Gazeta de informatica

Revista viitorilor specialiști în tehnologia informației

Ianuarie 2002
Vol. 12/1

Lei 20.000

din sumarul numărului:

Profil Apple Computer

Teoria fractală și geografia • Probleme-ACM 2002
Bursele Agora, rezolvările problemei de la prima rundă • Pentium 4 la 0.13 microni, Borland JBuilder 6

GHIDUL TĂU ÎN LUMEA TEHNOLOGIILOR

Misiunea noastră este de a testa și evalua produse și servicii din domeniul tehnologiei informației și comunicațiilor și de a prezenta rezultatele cu onestitate și obiectivitate

AGORA MEDIA Str. C-tin Rădulescu Motru nr. 13, etaj 4, apt. 42, sect. 4 (CP 94/49), București
Tel: 01-3309282, Fax: 01-3309285

www.pcmagazine.ro

Agora MEDIA

Vol. 12, Nr. 1 /2002

gazeta de informatică

Redacția:

Director științific:
prof. dr. *Horia Georgescu*
Redactor șef:
lect. univ. *Clara Ionescu*
Redactor șef adjunct:
Mihai Scorțaru
Redactori:
Carmen Șușman
Claudiu Soroiu
Au mai colaborat:
Cristian Kalanyos (Cluj)
Ciprian Ileană (Alba Iulia)
Mihai Stroe (București)
Tehnoredactare:
Claudiu Soroiu
Coperta și grafica:
Andrei Pașa
Adresa pentru corespondență:
Computer Libris Agora,
Universității 8/8,
3400, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 064/192.422
065/166.516
Fax: 065/166.290
http://www.ginfo.ro
ISSN: 1222-7129

Tiparul: *Master Druck* Târgu Mureș
tel: 065/162.359

Editor: Agora Media SRL
Agora Media SRL este membru al Biroului Român pentru Auditarea Tirajelor
Editor al publicațiilor "eWeek România", "PC Magazine România", "NetReport" și "Gazeta de Informatică"
str. T. Vladimirescu nr. 63/9
4300 Târgu-Mureș

Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director tehnic:
Szabó László
Vânzare reclamă:
Mihai Bucuroiu
Bogdan Manu
Florin Țiță
Denis Marioara
Marius Comărniceanu
Reprezentanța București:
Director: *Dana Pastramă*
Tel: 01/330.92.81
Fax: 01/330.92.85

Sumar

actual.. PAG. 5

Produsele lunii
Pentium 4 de 0,13µ
Borland JBUILDER 6
BLACK and WHITE
PRICE.RO

probleme.. PAG. 9

propuse pentru REZOLVARE

ACM 2002

~etapa sud-est europeană~
~București, România~

interviu.. PAG. 13

În dialog cu învingătorii

Mihai STROE

~impresii ale câștigătorilor~
~concursului ACM~

concurs.. PAG. 15

Rezolvările problemelor primei runde

Clasamentul general

Problemele rundei a patra a ediției 2001/2002

focus.. PAG. 24

Metode de SORTARE

Mihai SCORȚARU

~eficiența algoritmilor~
~SelectSort și InsertSort~

babel.. PAG. 32

Agenți INTELIGENȚI

Ciprian ILEANĂ

Teoria fractală și GEOGRAFIA

Cristian KALANYOS

profil.. PAG. 42

APPLE Computer

Claudiu SOROIU

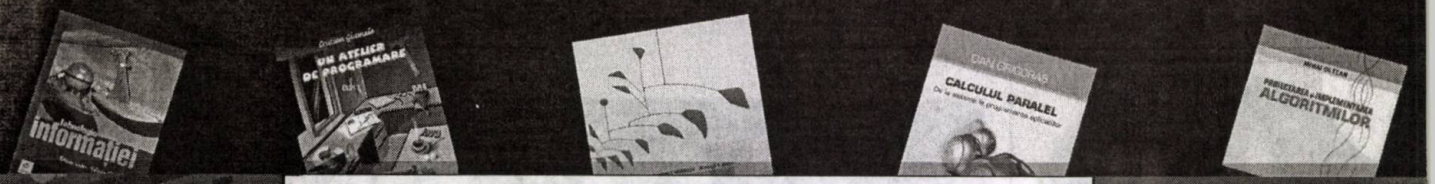
~istoria unei firme celebre~

enjoy.. PAG. 46

La închiderea EDIȚIEI

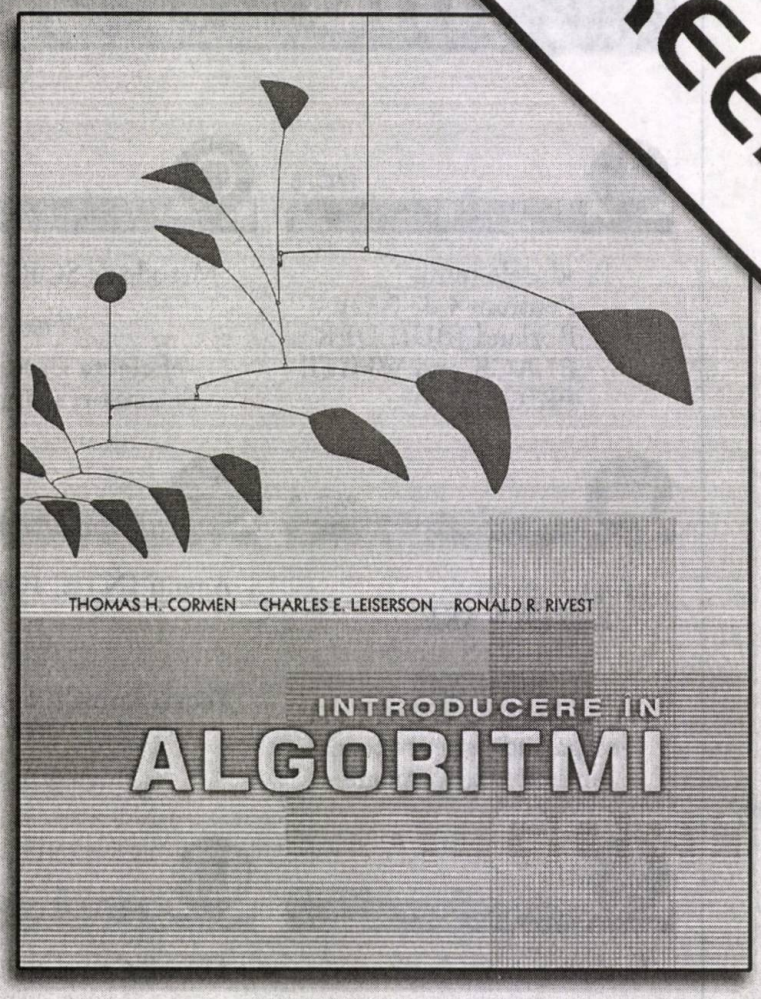
Carmen ȘUȘMAN

~evenimente "reale" care ~
~se petrec în redacția ~
~GInfo ~



COMPUTER LIBRIS AGORA

REEDITARE



Cartea se adresează, în primul rând, celor care studiază la nivel elementar sau academic algoritmi și structuri de date. Deoarece tratează aspecte teoretice și practice, legate de proiectarea algoritmilor, este la fel de binevenită elevilor, studenților, profesorilor și autodidacților.
Introducere în algoritmi acoperă atât metodele clasice cât și metodele moderne ca analiza amortizată și algoritmii paraleli.
Prezentarea matematică, riguroasă, este atent detaliată în așa fel încât să fie accesibilă tuturor cititorilor.
Capitolele încep cu elemente de bază, avansând spre subiecte mai complexe.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

"IARNĂ"... în România

Oare de ce am crezut întotdeauna că viața este mult mai frumoasă în "țările calde"? Să fie doar o întâmplare că am fost atrasă mereu de imaginea plajelor însorite dintr-un peisaj mediteranean cu căsuțe scăldate în soare sub un cer albastru și oameni bronzați și veseli?

Lăsând la o parte imaginile "simbolice" ale unei lumi visate, am putut constata "pe viu" că în Spania aș putea *trăi*, că în Cipru m-am *împrietenit* incredibil de repede cu toți oamenii pe care i-am întâlnit, cu vânzătorul de souveniruri, cu taximetristul, cu... nu are rost să continui... veți considera că nu am știut să mă comport rezervat și civilizat, așa cum i-ar fi stat bine unei persoane serioase de vârsta mea. Nu vreau să exagerez cu argumente și exemplificări. Aș vrea doar să încerc să găsesc o explicație valabilă pentru această simpatie a mea. Să fie mult mai simplu? Să fie doar Soarele "de vină"?

Am văzut cum e și în țările nordice. În Finlanda, tot ce este important se derulează în..., sau în jurul saunei. Alt loc "cald"...

La propriu și la figurat, noi oamenii căutăm căldura, ne simțim bine dacă este cald...

Gânduri de iarnă... veți zâmbi dezamăgiți. Dar nu vreau să țin nici o "predică". Aș vrea să auziți că țip disperată!

Ne plac țările calde pentru că avem impresia că acolo libertatea este *libertate*. În acea lume unde îți iei tricoul și sandalele și te duci să întâmpini o zi nouă, există mult mai puține bariere, mult mai puține minciuni între oameni.

Aici, în România, există pericolul ca iarna să se furișeze și în sufletele noastre. Și de data aceasta, mă refer la apatia generală despre care se vorbește, la neîncrederea în politicieni, la impertinența cu care cei acuzați de corupție răspund la întrebări... Dar la fel de grav mi se pare faptul că există medici care nu își fac datoria, există profesori care nu merg la ore sau merg nepregătiți...

În fond, ce vede azi un adolescent... și nu doar în România..., pentru că nu cred că țara noastră constituie o excepție? Deci, ce vede un copil, un tânăr?

În familie? Părinții sunt obosiți și nervoși pentru că aleargă pentru asigurarea acelui "trai decent", mult râvnit. La școală? Profesorii sunt plictisiți, blazați (excepțiile întăresc regula). Unii, discută despre problemele lor de existență, despre salariile lor mizerabile. De ce să caute să-și îmbunătățească stilul? Vorbesc despre stilul de predare, despre stilul de învățare... Da, unii învață chiar și după 10, 20, ... oare câți ani? Alții, dezamăgiți pe drept sau "în general", se ascund în spatele unor scuze transparente: "totul e degeaba". De ce, dragi colegi? De ce, dragi elevi? Când totul ar trebui să funcționeze în interesul *nostru*?

Nu vorbim de mari afaceri, de miliarde de lei sau milioane de dolari. Pe noi ne interesează *Școala*. Școala, unde copilul vine cu încredere și, din păcate, pleacă dezamăgit. Și i se întâmplă asta nu din cauza că habar n-am care politician este mincinos. Și totuși, lucrurile se leagă! Dacă unii nu ar fura milioane de dolari din bugetul statului, poate, poate... ar rămâne ceva și pentru învățământ...

Cum nu se vede, cum nu se știe, cum nu se ia la cunoștință? Dacă o țară nu investește în învățământ, adică în tineret, nu are șansa dezvoltării!

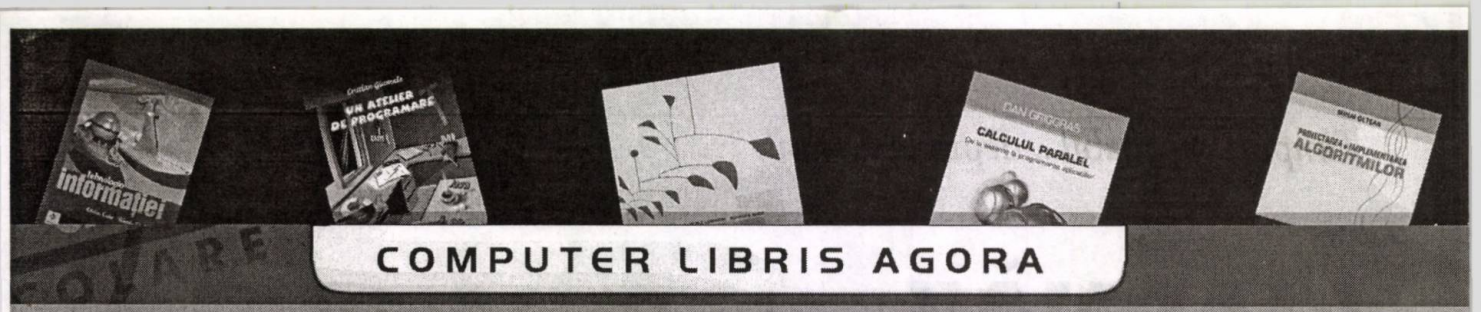
Chiar și în informatică, parcă totul merge "înapoi". Nu neapărat din cauză că sunt ore mai puține, ci mai degrabă din cauză că de acum încolo, în multe locuri, nu se va lucra pe semigrupe, ci cu clasa întreagă. Credeți că interesează pe cineva că informatica zilelor noastre nu se poate preda sau e inutil să se predea fără calculator?!

Unii copii învață din notițe, dictate de profesor din caietul lui... scris acum 10-20 de ani. Nu credeți? Rău e că exact ei, acești profesori și elevii lor nu citesc nici *GInfo*. Poate profu' face rost de câte un număr, dar cum să dea așa ceva în mâna copiilor? Este riscant... poate îl întreabă vreun năzdrăvan de copil de un subiect tratat în revistă. Sau, în cazul unui profesor mai curajos: "De ce să flutur gazeta? Deocamdată, aici, la această catedră, eu sunt șef! Uite ce problemă șmecheră v-am adus! Cine ce treabă are de unde?" Oare e tipic est-european?

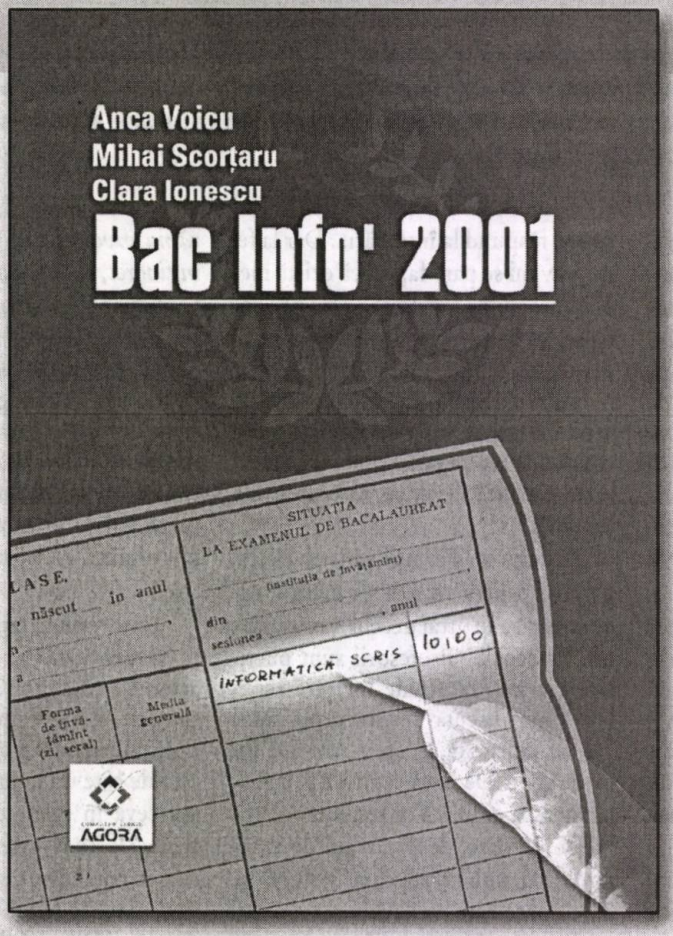
Poate e la fel în Ucraina, Moldova, Bulgaria, Serbia, Ungaria și... în altă parte. Dar în unele țări "mai calde" lumea știe să se bucure de o *fiesta*... Și e bine: fiesta este în fiecare săptămână. Se dansează, se cântă, lumea se îmbrățișează și se iubește. În țările calde oamenii se iubesc. Apropos, România nu ar putea să devină și ea "o țară caldă"?

Depinde de noi... Doar de noi!

Clara Ionescu



COMPUTER LIBRIS AGORA



Bac Info' 2001 conține 20 de variante de subiecte pentru Bacalaureat. Fiecare variantă conține câte patru subiecte. Primul dintre ele conține trei subpuncte care testează capacitățile candidaților de a recunoaște valoarea de adevăr a unei propoziții, de a determina efectul unui anumit algoritm și de a găsi o anumită greșeală într-un algoritm. Al doilea subiect conține doua subpuncte; primul testează deprinderea candidaților de a folosi limbajul pseudocod, iar al doilea cea de a folosi un limbaj de programare (Pascal sauC++). Al treilea subiect este o problemă pentru rezolvarea căreia este necesară folosirea unor structuri dinamice de date, iar al patrulea este o problemă pentru a cărei rezolvare trebuie folosită metoda backtracking.

COMPUTER LIBRIS AGORA

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

Pentium 4 de 0,13µ

Prima noutate tehnologică a anului 2002 este cea a fabricării noului procesor Pentium 4 oferit de Intel. Realizat pe baza unei tehnologii de 0,13µ, procesorul a devenit, încă de la lansare, procesorul cu cea mai mare frecvență de funcționare: 2,2 GHz. Datorită noutăților pe care le aduce, acest procesor primește titlul de produs al lunii ianuarie...

Introducere

Pe data de 7 ianuarie 2002 a fost lansată o nouă familie de procesoare *Intel Pentium 4*.

Această lansare marchează prima modificare semnificativă în tehnologia de fabricare a procesoarelor *Pentium 4*. De la lansarea sa, în noiembrie 2000, procesorul *Pentium 4* a fost fabricat pe baza unei tehnologii de 0,18µ. La început, aceste procesoare erau folosite împreună cu plăci de bază compatibile cu *Socket 423*. Singura modificare apărută în cursul anului 2001 a fost apariția unor versiuni care foloseau *Socket 478*.

Prima noutate semnificativă a apărut doar la începutul acestui an. Cunoscut până acum sub numele de cod *Northwood*, noul *Pentium 4* înlocuiește tehnologia de fabricare de 0,18µ cu cea de 0,13µ.

Tehnologia de 0,13µ

Noul *Pentium 4* nu este primul procesor fabricat pe baza unei tehnologii de 0,13µ.

În vara anului trecut au apărut primele procesoare *Pentium III* care foloseau această tehnologie și erau cunoscute cu numele de cod *Tualatin*. Acestea au apărut în versiuni pentru *server*-e, *desktop*-uri și calculatoare portabile.

La sfârșitul anului 2001 s-a trecut la folosirea tehnologiei de 0,13µ și pentru ultimele procesoare din familia *Celeron*.

Date tehnice

Procesorul *Pentium 4* fabricat pe baza tehnologiei de 0,13µ a apărut în două variante: una cu frecvența de 2 GHz și una cu frecvența de 2,2 GHz.

Folosirea tehnologiei de 0,13µ a permis reducerea dimensiunii procesorului de la 217 mm^2 la 146 mm^2.

Numărul tranzistorilor folosiți pentru fabricarea procesorului a crescut de la 42 de milioane la 55 de milioane.

Nu există decât variante pentru *Socket 478*, procesorul neputând fi utilizat pe plăci de bază cu *Socket 423*.

Frecvența magistralei sistem este de 400 MHz, la fel ca cea a celorlalte procesoare *Pentium 4*, dar se preconizează în curând apariția unei versiuni care să folosească o frecvență de 533 MHz pentru magistrala sistem.

Dimensiunea celui de-al doilea nivel al memoriei *cache* s-a dublat de la 256 KB la 512 KB.

Plăcile de bază compatibile cu acest procesor trebuie să folosească *chipset*-uri *Intel 845 (D)*, *Intel 850*, *VIA P4X266*, *VIA P4X266A* sau *SiS 645*.

Tipurile de memorii care pot fi utilizate împreună cu acest procesor sunt *SDRAM*, *DDR-SDRAM* și *RDRAM*.

Noul *Pentium 4* oferă, la fel ca și predecesorul său, suport pentru familiile de instrucțiuni *MMX*, *SSE* și *SSE2*, dar nu și pentru *3DNow!* și *3DNow! Professional*.

Valoare tensiunii folosite a fost redusă de la 1,7 V la 1,5 V.

Performanțe

Este cunoscut faptul că o frecvență de ceas superioară nu implică și o performanță mai mare.

După lansarea procesorului *Pentium 4* cu frecvența de 2 GHz, acesta era comparat cu cel mai bun procesor rival, *AMD Athlon XP 1900+*, a cărui frecvență era de doar 1,6 GHz.

În ciuda diferenței de 400 MHz dintre cele două procesoare, *Athlon XP 1900+* s-a dovedit superior unui *Pentium 4* cu frecvența de 2 GHz în majoritatea *benchmark*-urilor.

Acum, cel mai performant procesor produs de *AMD* este *Athlon XP 2000+* și lucrează la frecvența de 1,67 GHz, dar nu aduce nici o noutate tehnologică.

Între cel mai performant procesor *Intel* și cel mai performant procesor *AMD* există o diferență de 533 MHz. În plus, noutățile tehnologice oferă încă un avantaj procesorului *Intel*. Chiar și așa, un *Pentium 4* la 2,2 GHz nu depășește în toate *benchmark*-urile un *Athlon XP 2000+* sau un *Athlon XP 1900+*. Există unele *benchmark*-uri în care chiar și un *Athlon XP 1700+* depășește noul procesor.

Concluzie

Noul procesor introduce o serie de îmbunătățiri care contribuie la creșterea performanțelor. Totuși, noul *Pentium 4* nu a devenit lider de necontestat în acest domeniu.

Borland JBUILDER 6

JBuilder este unul dintre cele mai cunoscute și mai performante medii de programare Java. La sfârșitul anului 2001 compania Borland Software a lansat versiunea cu numărul 6 al acestui produs. Datorită îmbunătățirilor și facilităților introduse, Borland JBuilder 6 a devenit produsul lunii ianuarie.

Prezentare

De curând, compania *Borland* a lansat pe piață un nou produs și anume *JBuilder*, ajuns acum la versiunea 6. Mediul de programare *JBuilder 6* este proiectat pentru a veni în ajutorul programatorilor în *Java*.

Sigur că vă veți pune întrebarea "De ce *Java*?". Răspunsul la această întrebare este foarte simplu: *Java* este un limbaj de programare care îmbină foarte multe dintre facilitățile limbajelor de programare de nivel înalt, ceea ce îl face utilizabil în foarte multe domenii cum sunt aplicațiile *client/server*, suport interactiv pentru *server*-e *Web*, *e-commerce*, baze de date etc.

JBuilder 6 include o serie de îmbunătățiri menite să ușureze munca programatorilor. Acest produs este disponibil în patru variante, și anume *JBuilder 6 Personal*, *JBuilder 6 Professional*, *JBuilder 6 Enterprise* și *Enterprise Studio, Java Edition*.

Pachetul *JBuilder 6 Personal* nu poate fi folosit pentru crearea de aplicații comerciale, ci este destinat folosirii în scopuri didactice. Pentru a avea dreptul să utilizați în scopuri comerciale aplicațiile pe care le creați, va trebui să folosiți *JBuilder 6 Professional*, *JBuilder 6 Enterprise* sau *Enterprise Studio, Java Edition*.

Mediul de programare *JBuilder 6* este un mediu vizual pentru crearea de aplicații. Acesta oferă suport pentru ultimele standarde *Java*. *JBuilder 6* este implementat în întregime în *Java* pentru o foarte bună interoperabilitate și performanță pe platforme ca: *Windows*, *Linux*, *Solaris*, *Mac OS* și toate celelalte sisteme de operare care au suport pentru *Java 2 SDK 1.3*.

Cu ajutorul editorului vizual integrat, se pot crea diagrame *UML* (*Unified Modeling Language*), care vă ajută să observați mai ușor interdependența claselor.

Sistemul necesar

Pentru a putea fi folosit optim, *JBuilder 6* necesită 256 MB RAM, 700 MB spațiu liber pe disc, o unitate *CD-ROM*, o placă video care să suporte o rezoluție de 800 × 600 cu cel puțin 256 de culori, și un *mouse* sau orice alt dispozitiv similar.

Dacă pe calculatorul vostru rulează unul din sistemele de operare *Microsoft Windows* XP, *Microsoft Windows 2000* (cu *Service Pack* 2), sau *Microsoft Windows NT* 4.0 (cu *Service Pack* 6), aveți nevoie de un procesor *Intel Pentium II* care lucrează la o frecvență de cel puțin 233 MHz sau unul echivalent.

Pentru un sistem pe care rulează sistemul de operare *Linux* aveți nevoie de același tip de procesor ca în cazul sistemului de operare *Windows*, dar și de o interfață grafică.

Pentru sistemele de operare *Solaris 7* (2.7) sau *Solaris 8* (2.8) aveți nevoie de un procesor *UltraSPARC II* sau mai bun.

În cazul în care folosiți o platformă *MacIntosh* aveți nevoie de sistemul de operare *Apple Mac OS X* și de un procesor *G3* care rulează la frecvența de cel puțin 350 MHz.

Descrierea pachetelor

JBuilder 6 Personal este destinat celor care doresc să învețe limbajul *Java* și să se familiarizeze cu mediul de programare oferit de *Borland*. Pachetul include un editor, un depanator, un compilator, diferite exemple, precum și diferite alte instrumente ajutătoare.

JBuilder 6 Professional vă pune la dispoziție toate instrumentele oferite de varianta *Personal*, la care se adaugă facilități pentru crearea de aplicații *Web* sau cu baze de date etc.

JBuilder 6 Enterprise adaugă la facilitățile oferite de variante *Professional* câteva elemente noi, cum ar fi vizualizarea codului *UML*.

În sfârșit, *Enterprise Studio, Java Edition* oferă, în plus față de varianta *Enterprise*, pachetele *Rational Rose* cu integrare pentru *JBuilder*, *Rational Unified Process* și *Macromedia Dreamweaver UltraDev*.

Încheiere

Compania *Borland* oferă, în acest moment, câteva pachete performante destinate programatorilor. Alături de *JBuilder 6*, aceștia au la dispoziție medii de programare vizuale pentru limbajele *Pascal* (*Borland Delphi 6*) și *C++* (*Borland C++ Builder 5*).

Totuși, pachetul *Microsoft Visul Studio 6.0* reprezintă o alternativă viabilă la produsele *Borland*.

BLACK and WHITE

~ Creature Isle ~

Jocul Black and White, realizat de către Lionhead Studios sub îndrumarea celebrului designer Peter Molineaux și distribuit de Electronic Arts a introdus câteva concepte revoluționare în domeniul jocurilor pe calculator. La mijlocul lunii ianuarie apare un add-on numit Creature Isle. Chiar dacă nu este un joc nou, acest produs merită titlul de joc al acestei luni.

Despre Black and White

După câțiva ani de așteptare, la începutul lunii aprilie a anului 2001 a apărut pe piață jocul *Black and White*. Acest joc îmbină, într-o manieră excepțională, majoritatea tipurilor de jocuri având elemente de strategie de construcții, acțiune, aventură etc.

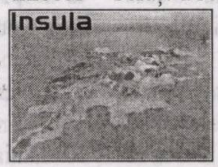
În *Black and White* intrați în rolul unui zeu care stăpânește un popor. Sunteți singurul care decide dacă veți fi un zeu bun, iubit de locuitori, sau unul malefic, care domină cu ajutorul terorii.

Pe parcursul celor cinci niveluri ale jocului veți avea de îndeplinit o serie de sarcini până în momentul confruntării finale cu *Nemesis*, zeul inamic.

Pe parcursul jocului sunteți ajutat de o creatură care este reprezentantul dumneavoastră în lumea muritorilor.

Jocul este deosebit de complex, existând câteva sarcini principale care trebuie neapărat îndeplinite și în jur de 100 de sarcini secundare după îndeplinirea cărora veți obține diferite recompense. Va trebui să vă îngrijiți de sătenii dumneavoastră pentru ca aceștia să vă venereze și astfel să vă puteți folosi calitățile magice. Un rol foarte important îl are creatura. Aceasta încearcă să vă imite acțiunile și, dacă aveți grijă de ea, va învăța să se comporte așa cum doriți. Ea se va lupta cu diferite alte creaturi și voi veți avea un rol pasiv în aceste confruntări. De asemenea, aria voastră de influență este limitată, iar pentru a acționa în exteriorul acesteia va trebui să vă încredeți în forțele creaturii.

La început puteți alege între trei creaturi: tigru, vacă sau maimuță. Pe parcursul jocului veți putea înlocui creatura aleasă cu o oaie, un urs brun, un urs polar, un lup, un cal, o panteră, o broască țestoasă și multe altele.



Creature Isle

Un grup de marinari pe care i-ați ajutat în primul nivel au descoperit în călătoriile lor o nouă insulă pe care se află o mulțime de creaturi.

După ce l-ați învins pe *Nemesis*, nu aveți altceva de făcut decât să explorați această insulă. Aici veți găsi o breaslă a creaturilor și scopul principal este ca creatura voastră să fie acceptată în rândul breselei.

Fiecare dintre cele aproximativ 25 de creaturi pe care le veți găsi vă va cere să duceți la îndeplinire o anumită sarcină.

După îndeplinirea sarcinii, creatura respectivă o poate înlocui pe cea pe care o aveți în acel moment.

În *Black and White* creaturile sunt considerate de genul masculin, deci în timpul jocului se fac referiri la creatură folosindu-se pronumele ***el*** (engl. ***he***). Evident trebuie să existe și o femelă, iar aceasta este păzită de un crocodil, liderul frăției creaturilor.

Dacă creatura va reuși să intre în grațiile *Evei*, creatura femelă, atunci în scurt timp va apărea o creatură bebeluș. Totuși, până atunci va trebui ca creatura voastră să își demonstreze calitățile de părinte.

Un rol deosebit de important îl are, în această situație, *Tyke*. Această rățușcă va deveni discipolul creaturii voastre și va învăța de la aceasta așa cum creatura a învățat de la voi.

Dacă *Eve* va fi mulțumită de modul în care a fost antrenat *Tyke*, atunci există șansa ca creatura voastră să întemeieze o familie.

După îndeplinirea tuturor sarcinilor, creatura voastră devine membru al frăției și va putea purta însemnul acesteia în confruntările *online* cu alți jucători.

Creature Isle introduce, pe lângă numărul mare de creaturi noi, câteva vrăji, dar îmbunătățește semnificativ *AI*-ul creaturii care știe să ducă la bun sfârșit o mulțime de acțiuni noi.

PRICE.RO

La www.price.ro veți găsi unul dintre cele mai bine realizate site-uri de informații din domeniul IT. Chiar dacă numele sugerează faptul că site-ul este dedicat prețurilor diferitelor produse, conținutul său este mult mai bogat. Calitatea acestui site ne-a determinat să îi acordăm titlul de site al lunii ianuarie.

Istoric

În vara anului trecut, pe unul dintre cele mai cunoscute forumuri de discuție din România, autorul *site*-ului anunța lansarea primului *site* românesc dedicat celor care doresc să achiziționeze diferite componente *hardware*.

Site-ul prezenta ofertele celor mai cunoscuți distribuitori români și, în scurt timp, a devenit un *site* de referință pentru cei interesați de prețurile de pe piața românească.

În numărul 11/6 (octombrie 2001) am realizat o scurtă prezentare a acestui *site* în care descriam facilitățile oferite în acel moment.

Între timp lucrurile s-au schimbat. Numărul celor care accesează paginile acestui *site* a crescut foarte repede, deoarece există foarte mulți utilizatori care sunt interesați de informațiile oferite. În revistele IT românești au apărut o mulțime de prezentări în care erau apreciate calitățile acestui *site*.

Ca urmare, autorul a decis că ar fi indicat să își dezvolte *site*-ul și acum prețurile componentelor reprezintă doar o secțiune a *site*-ului.

Complexitatea *site*-ului a crescut și acum există o mulțime de colaboratori care se ocupă cu realizarea acestuia.

Noul price.ro

Dacă în vara anului trecut acest *site* oferea doar liste de prețuri și informații de contact pentru diferite firme, acum conținutul său este mult mai diversificat. Există mai multe secțiuni printre care una dedicată știrilor de ultimă oră, o secțiune dedicată articolelor IT, precum și una dedicată testelor.

Vi se pune la dispoziție o secțiune de *download* prin intermediul căreia puteți descărca diferite programe gratuite destinate unei game variate de utilizatori.

O secțiune foarte interesantă a *site*-ului este forumul de discuții care are diferite secțiuni. Aici puteți discuta despre diferite subiecte sau puteți cere ajutorul cunoscătorilor în diferite probleme cu care vă confruntați. Trebuie menționat faptul că forumul este unul dintre cele mai des vizitate din România și aceasta se datorează, în mare măsură, moderatorilor care nu abuzează de drepturile lor, dar nici nu permit apariția de texte care să încalce politica *site*-ului.

Secțiunea de știri conține ultimele noutăți din domeniu. Știrile sunt, cu adevărat, știri de ultimă oră și sunt prezentate într-o manieră profesionistă. Din acest punct de vedere *site*-ul poate rivaliza cu orice *site* de acest gen de peste hotare și, cu siguranță, ocupă un loc fruntaș într-un eventual clasament al *site*-urilor românești de acest tip. Există o mulțime de categorii de știri cum ar fi *hardware*, *software*, *jocuri* etc.

În situația puțin probabilă în care sunteți interesați de platformele celor care vizitează acest *site*, aveți la dispoziție o secțiune în care sunt prezentate diferite statistici referitoare la vizitatori.

Concluzie

Site-ul **price.ro** este unul dintre cele mai bine realizate *site*-uri românești și demonstrează faptul că o simplă idee, dacă este bine pusă în practică, poate duce la obținerea unor rezultate spectaculoase.

Nu putem decât să vă recomandăm să vizitați acest *site*, deoarece veți găsi aici o mulțime de informații utile și interesante.

Trebuie, totuși, să vă avertizăm că acest *site* a fost creat pentru a fi vizualizat folosindu-se programul de navigare *Internet Explorer* (versiunea 4 sau o versiune mai nouă) și necesită o rezoluție de cel puțin 1024 × 768 pentru ca paginile să fie afișate corect. În plus, există unele probleme la afișarea meniurilor care pot duce la apariția unor situații bizare.

În final, încă un avertisment... Există șanse, chiar destul de mari, ca unele prețuri publicate pe acest *site* să nu mai fie actuale.

Prețurile sunt preluate din ultimele oferte ale distribuitorilor și pot fi modificate fără ca aceste schimbări să apară imediat pe *site*.

De aceea, vă recomandăm să verificați dacă prețul indicat pe *site* corespunde într-adevăr ultimei oferte a distribuitorului.

București, România

PROBLEME propuse pentru REZOLVARE

Vă prezentăm enunțurile problemelor de la faza regională pentru sud-estul Europei a concursului ACM. Echipele universităților participante s-au confruntat cu opt probleme ale căror enunțuri le puteți citi în paginile următoare.

P010201: Polinom optim

Se consideră un număr întreg pozitiv K și un șir S care este format din cifre. Șirul S poate fi partiționat în mai multe șiruri nevide S_0, S_1, S_2, ... astfel încât $S_0 + S_1 + S_2 + ... = S$, unde + reprezintă operația de concatenare.

Fiecare partiție a lui S care este formată din M subșiruri definește un polinom $P(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + ... + a_M \cdot x^M$, unde a_i este numărul reprezentat de subșirul S_i.

De exemplu, șirul $S = 1204$ poate fi partiționat în $S = 1 + 204$ (polinomul $P(x) = 1 + 204 \cdot x$), în $S = 1 + 2 + 04$ (polinomul $P(x) = 1 + 2 \cdot x + 4 \cdot x^2$) etc.

Trebuie să scrieți un program care alege, dintre toate polinoamele care pot fi construite pe baza șirului S, polinomul pentru care valoarea $P(K)$ este minimă.

Se știe că valoarea maximă a rezultatului nu depășește $12 \cdot 10^{14}$.

Date de intrare

Fișierul de intrare **A.DAT** conține pe prima linie numărul n de perechi (K, S) pentru care se va construi polinomul optim.

Pe fiecare din următoarele n linii se află două numere separate printr-un singur spațiu; primul număr este K, iar al doilea este șirul de cifre S.

Date de ieșire

La ieșirea standard se vor scrie n linii. Pe fiecare linie se va scrie polinomul optim corespunzător testului din fișierul de intrare.

Exemplu

A.DAT

```
3
1 1234
3 123
1 1001
```

Ecran

```
1 + 2 * X^1 + 3 * X^2 + 4 * X^3
12 + 3 * X^1
1 + 1 * X^1
```

P010202: Soldați

Soldații armatei selenare sunt aliniați; unii privesc spre capătul din dreapta al liniei, iar ceilalți privesc spre capătul din stânga.

Așadar, anumite perechi de soldați aflați în linie pot sta *față în față*, anumite perechi pot sta *spate în spate*, iar altele *față în spate* (echivalent cu *spate în față*). Se dorește să nu existe perechi de soldați care stau *față în față*.

Pentru aceasta aveți la dispoziție un singur tip de comandă prin care puteți cere unei perechi de soldați care stau *față în față* să se întoarcă. După întoarcere, cei doi soldați vor sta *spate în spate*.

Pentru o aliniere dată a soldaților, trebuie să verificați dacă există o succesiune de comenzi în urma căreia să nu existe nici o pereche de soldați care să stea *față în față*. Dacă acest lucru este posibil, va trebui să determinați numărul minim de comenzi necesare.

Date de intrare

Fișierul de intrare **B.DAT** conține pe prima linie numărul t al seturilor de date ($0 < t < 10$). Următoarele linii conțin datele de test. Prima linie corespunzătoare unui test conține un număr natural k ($0 < k < 1000$) care indică numărul liniilor următoare corespunzătoare testului curent. Aceste linii descriu pozițiile soldaților; pe fiecare linie se află cel mult 50 de caractere. Pozițiile soldaților sunt indicate de simbolurile '<' (soldatul privește spre stânga) și '>' (soldatul privește spre dreapta).

Date de ieșire

La ieșirea standard se vor scrie t linii corespunzătoare celor t seturi de date. Dacă este posibilă aranjarea soldaților

astfel încât să nu existe nici o pereche de soldați care să stea *față în față*, pe linia corespunzătoare se va scrie numărul minim de întoarceri necesare. Dacă aranjarea nu este posibilă, pe această linie se va scrie valoarea -1.

Exemplu

B.DAT

```
3
2
><>
<
1
><<<<
3
<
<
>
```

Ecran

```
3
4
0
```

P010203: Numere secrete

Se aleg două numere naturale a și b astfel încât $1 < a < b$. Unei persoane P i se comunică produsul celor două numere ($a \cdot b$), iar unei persoane S i se comunică suma celor două numere ($a + b$). Are loc următoarea discuție între P și S:

P: Nu știu care sunt numerele a și b!
S: Nici eu nu le știu, dar știam că tu nu le poți ști!
P: Acum știu numerele!
S: Și eu le știu acum!

Date de intrare

Fișierul de intrare **C.DAT** conține perechi de numere x și y ($2 \le x < y \le 550$), câte una pe o linie. Se garantează corectitudinea intrării.

Date de ieșire

Pentru fiecare pereche furnizată la intrare, trebuie găsite toate perechile de numere a și b ($x \le a < b \le y$) pentru care discuția dată este posibilă.

Perechile corespunzătoare unui interval dat la intrare vor fi scrise pe aceeași linie. Dacă există astfel de perechi, linia se va încheia cu textul `"no more pairs."`. Dacă nu există nici o astfel de pereche, linia va conține doar textul `"no pairs."`.

Numerele care formează o pereche vor fi separate prin caracterul `','`, iar perechile vor fi separate între ele prin caracterul `';'`, urmat de un spațiu.

Exemplu

C.DAT

```
2 10
2 20
```

Ecran

```
no pairs.
4,13; no more pairs.
```

P010204: Tipografie

Directorii unei tipografii foarte mari (cu resurse nelimitate) doresc să știe care este timpul minim necesar pentru a satisface cerințele unui anumit client. O comandă constă în cererea de tipărire a unui anumit număr de reviste, pliante, cărți etc. Pentru fiecare produs care este tipărit, trebuie parcurs un șir de pași. De exemplu, prima mașină preia o foaie de hârtie dintr-un teanc, a doua taie hârtia pentru ca aceasta să aibă dimensiunile dorite, alta tipărește pagina, alta pune paginile tipărite într-un alt teanc, iar ultima lipește paginile din acest teanc. Acest flux de tipărire poate fi descris dându-se cele N operații necesare, timpul consumat de fiecare dintre acestea și dependențele care există între operații (de exemplu, operațiile 0, 1 și 2 pot începe imediat, operația 3 poate începe doar după terminarea operației 2, operația 4 poate începe după terminarea operațiilor 1 și 3, operația 5 poate începe după terminarea operației 1, iar operația 6 poate începe după terminarea operațiilor 0, 4 și 5). Trebuie să scrieți un program care determină timpul minim necesar pentru a satisface cerințele clientului.

Date de intrare

Fișierul de intrare **D.DAT** conține mai multe seturi de date, separate printr-o linie vidă. Fiecare set de date reprezintă comanda unui client și are următoarea structură:

- pe prima linie se află numărul N (cel mult 100) al operațiilor necesare pentru a satisface comanda;
- pe fiecare dintre următoarele N linii se află informații referitoare la o operație; primul număr de pe linie este codul de identificare al operației (cuprins între 0 și N - 1), al doilea (un întreg strict pozitiv) reprezintă timpul necesar pentru îndeplinirea operației, iar următoarele reprezintă codurile de identificare ale operațiilor care trebuie terminate înainte ca operația curentă să poată începe; ultimul număr al liniei este un terminator cu valoarea -1.

Date de ieșire

Pentru fiecare set de date se va tipări la ieșirea standard câte o linie care va conține timpul minim necesar pentru satisfacerea comenzii. În cazul în care comanda nu poate fi satisfăcută, se va scrie valoarea -1.

Exemplu

D.DAT

```
3
0 2 1 2 -1
1 3 -1
2 2 -1

2
0 1 1 -1
1 1 0 -1
```

Ecran

```
5
-1
```

P010205: Permutare

Pentru o permutare $A = \{a_1, a_2, ..., a_n\}$ a mulțimii $\{1, 2, ..., n\}$ definim o secvență $P(A) = (p_1, p_2, ..., p_{n-1})$ unde $p_i = 0$ dacă $a_i > a_{i+1}$ și $p_i = 1$ dacă $a_i > a_{i+1}$. Se dă o permutare B și se cere determinarea numărului permutărilor C pentru care $P(B) = P(C)$, incluzând permutarea B.

Date de intrare

Fișierul de intrare **E.DAT** conține, pe câte o linie, mai multe seturi de date. Primul număr de pe linie reprezintă valoarea n și este urmat de alte n numere care reprezintă permutarea B. Toate numerele de pe o linie sunt separate prin câte un spațiu. Ultima linie a fișierului conține doar valoarea 0 și nu trebuie procesată.

Date de ieșire

La ieșirea standard se va scrie câte o linie pentru fiecare set de date. Fiecare dintre acestea va conține numărul permutărilor C corespunzătoare unei permutări date la intrare.

Exemplu

E.DAT

```
2 1 2
4 1 3 2 4
0
```

Ecran

```
1
5
```

P010206: Găsirea numărului

Se dă o mulțime S de numere întregi nenegative în care un anumit număr întreg poate apărea în S de mai multe ori și un număr întreg pozitiv k. Fie m cel mai mare număr întreg mai mic sau egal cu k pentru care există o sumblțime a lui S ale cărei elemente au suma m.

Problema determinării numărului m dacă sunt cunoscute S și k, este foarte dificilă.

Sarcina voastră va fi puțin mai simplă. Va trebui să determinați un număr întreg p a cărui valoare este cu cel mult 10% mai mică decât m. Mai exact, numărul p trebuie să respecte inegalitatea $m \cdot 0.9 < p \leq m$.

Date de intrare

Prima linie a fișierului de intrare **F.DAT** conține numărul seturilor de date. Fiecare set de date este descris prin trei linii. Pe prima dintre ele se va afla numărul n ($1 \leq n \leq 500$) al elementelor din mulțimea S, pe a doua linie se vor afla cele n elemente (numere întregi nenegative) ale mulțimii, separate prin câte un spațiu, iar pe a treia linie se va afla numărul k ($1 \leq k \leq 2000000000$). Nu există linii vide între două seturi de date consecutive.

Date de ieșire

La ieșirea standard se va scrie câte o linie pentru fiecare set de date. Fiecare dintre acestea va conține un singur număr întreg p care satisface inegalitatea $m \cdot 0.9 < p \leq m$. Nu trebuie să apară nici un spațiu la începutul liniilor.

Exemplu

F.DAT

```
2
5
10 4 5 6 4
24
4
3 8 2 11
10
```

Ecran

```
22
10
```

P010207: Monede contrafăcute

Banca Populară "Marile Speranțe" verifică monedele pe care le are în seif. Monedele ar trebui să aibă aceeași greutate și sunt așezate în grămezi. *Serviciul Secret* a furnizat o informație potrivit căreia ***exact o monedă din fiecare grămadă a fost înlocuită cu o monedă contrafăcută care este mai grea sau mai ușoară decât celelalte monede.*** Considerați această informație ca fiind veritabilă și scrieți un program care verifică grămezile.

Pentru fiecare grămadă și o serie de măsurători, programul trebuie să încerce să identifice moneda contrafăcută și greutatea sa relativă.

Date de intrare

Fiecare linie din fișierul **G.DAT** conține informații referitoare la o grămadă și la măsurătorile efectuate asupra monedelor din grămadă.

Formatul unei linii este:

$$N\ M\ măsurătoare_1\ ...\ măsurătoare_M$$

unde N ($3 \leq N \leq 1000$) este numărul monedelor din grămadă, iar M (număr pozitiv) este numărul măsurătorilor efectuate.

O măsurătoare are forma:

$monedă_1\ ...\ monedă_{w/2}\ indicator\ monedă_{w/2+1}\ ...\ monedă_w$,

unde w (număr par strict pozitiv) este numărul monedelor plasate pe talerele balanței, $monedă_i$ reprezintă o monedă amplasată pe taler ($1 \leq monedă_i \leq 1000$, $i = 1..w$, $monedă_i \neq monedă_j$ pentru $i \neq j$), iar indicator este unul din caracterele \, | sau / și are următoarea semnificație:

- \ - balanța se înclină spre stânga;
- | - balanța este echilibrată;
- / - balanța se înclină spre dreapta.

Elementele din fișierul de intrare sunt separate prin cel puțin un spațiu. Datele din fișier respectă în totalitate aceste specificații, dar pot fi inconsistente sau incomplete pentru a putea determina soluția.

Date de ieşire

La ieşirea standard se va scrie câte o linie pentru fiecare set de date. Acestea pot avea unul dintre următoarele formate:

- `Coin # is light` - moneda cu numărul # este contrafăcută şi este mai uşoară;
- `Coin # is heavy` - moneda cu numărul # este contrafăcută şi este mai grea;
- `Inconsistent data` - informaţia furnizată de *Serviciul Secret* este contrazisă sau folosind anumite măsurători date la intrare, se ajunge la concluzii contradictorii;
- `Incomplete data` - datele de intrare nu furnizează informaţii suficiente pentru a găsi moneda contrafăcută şi greutatea relativă a acesteia.

Exemplu

Cazurile din exemplul următor prezintă câteva situaţii de date de intrare corecte, incomplete sau inconsistente.

Pentru primul set de date avem trei monede şi două măsurători. La măsurătoarea `1 | 2` se cântăresc monedele 1 şi 2. Datorită faptului că balanţa este echilibrată, ambele monede sunt veritabile. La măsurătoarea `1 \ 3` sunt cântărite monedele 1 şi 3. Balanţa se înclină spre stânga, deci moneda 3 este mai uşoară.

Pentru al doilea set de date (3 monede şi două măsurători) este contrazisă informaţia furnizată de *Serviciul Secret* deoarece rezultă că toate cele trei monede sunt veritabile.

În cazul celui de-al treilea set de date măsurătorile duc la conluzii contradictorii:

- în urma primei măsurători tragem concluzia că monedele 1 şi 2 sunt veritabile;
- în urma celei de-a doua măsurători tragem concluzia că una dintre monedele 1 şi 2 este contrafăcută.

Pentru al patrulea set de date, putem trage concluzia că moneda 3 este contrafăcută, dar nu putem determina dacă ea este mai uşoară sau mai grea decât celelalte.

Pentru ultimul test, ştim că una dintre monedele 3 şi 4 este mai uşoară, dar nu ştim care dintre ele.

G.DAT

```
3 2   1 | 2   1 \ 3
3 2   1 | 2   1 | 3
3 2   1 | 2   1 \ 2
3 1   1 | 2
4 2   1 | 2   1 2 \ 3 4
```

Ecran

```
Coin 3 is light
Inconsistent data
Inconsistent data
Incomplete data
Incomplete data
```

P010208: Joc

Brian este un entuziast al jocurilor pe calculator, în special al celor care simulează realitatea virtuală. Acum, el se află în faţa *Porţii Stelare*. Pentru a putea deschide poarta, el trebuie să spargă protecţia cât mai curând posibil. Pentru spargerea protecţiei, el are nevoie de un cod (o secvenţă de numere) care să se potrivească cu secvenţa de numere care reprezintă protecţia *Porţii Stelare*. Această secvenţă de numere este foarte lungă; poziţia de start a primei apariţii a codului în secvenţa de numere a protecţiei va deschide poarta; îl puteţi ajuta?

Codul este o secvenţă de cel mult 60000 de numere întregi cuprinse între 0 şi 255. Programul va trebui să găsească poziţia primei potriviri (dacă există cel puţin o potrivire) sau să raporteze absenţa unei potriviri.

Date de intrare

Fişierului de intrare **H.DAT** conţine mai multe seturi de date. Fiecare set de date este descris prin patru linii. Pe prima dintre ele se va afla numărul *code_dimension* care reprezintă lungimea codului. Pe cea de-a doua linie se află numerele întregi care reprezintă codul. A treia linie conţine numărul *protection_dimension* care reprezintă dimensiunea protecţiei. Pe ultima linie corespunzătoare unui set de date se află numerele întregi care reprezintă protecţia. Datele de pe aceeaşi linie a fişierului de intrare sunt separate prin unul sau mai multe spaţii. De asemenea, pot apărea spaţii la începutul şi/sau la sfârşitul anumitor linii ale acestuia, sau linii vide sau care conţin doar spaţii. Acestea trebuie ignorate.

Date de ieşire

La ieşirea standard se va scrie câte o linie pentru fiecare set de date. În cazul în care codul se potriveşte cu protecţia, pe linia corespunzătoare se va scrie poziţia primei apariţii a codului în protecţia porţii. Numerotarea poziţiilor începe de la 0. Dacă nu există nici o potrivire, pe linia corespunzătoare se va scrie mesajul `no solution`.

Exemplu

H.DAT

```
3
0 1 2
7
2 3 4 0 1 2 5

2
1 4
6
4 1 4 1 4 4

3
1 2 3
7
3 2 1 3 2 255 7
```

Ecran

```
3
1
no solution
```

Despre ACM

Mihai Stroe

Faza regională a concursului ACM a fost câștigată de o echipă a Universității Politehnica din București. Echipa a fost formată din Gheorghe Andrei, Mugurel Ionuț Andreica și Florin Ghețu. Am adresat câștigătorilor câteva întrebări referitoare la acest concurs și nu numai. Le mulțumim celor trei pentru amabilitatea cu care au răspuns la aceste întrebări.

Pentru început permiteți-mi să vă felicit pentru câștigarea etapei regionale a concursului ACM. Cum vă simțiți în postura de învingători?

Suntem foarte bucuroși. Ne-am dorit foarte mult să fim primii și iată că am reușit să întrecem echipele tuturor universităților din sud-estul *Europei* care au participat la acest concurs.

Ce părere aveți despre modul în care a fost organizat acest concurs?

Nu avem nimic de reproșat organizatorilor. Din contră, totul s-a desfășurat în condiții foarte bune, fără incidente sau accidente.

Cum vi s-au părut cele opt probleme propuse spre rezolvare?

Dacă ne-ați fi întrebat acest lucru după prima citire a enunțurilor, v-am fi răspuns că șase din cele opt probleme sunt destul de dificile. Acum însă, când cunoaștem toate cele opt soluții, nu mai putem afirma același lucru. Problemele au fost, totuși, interesante și ne-au solicitat atât imaginația, cât și cunoștințele de algoritmică.

Ați fost primii în clasament de la început, după ce ați rezolvat prima problemă, și până la sfârșit dar, în ultima jumătate de oră, nu s-a mai anunțat evoluția echipelor. Rezultatul final a devenit public, ca de obicei, la festivitatea de premiere. V-a fost teamă de o răsturnare a situației?

Înainte de anunțarea echipelor care s-au clasat pe primele două locuri (știam că vom fi una din primele două echipe), eram neliniștiti. Apoi, când am aflat că pe locul II s-a clasat echipa din *Sofia*, i-am aplaudat foarte tare pe cei trei membri ai acestei echipe, care au fost foarte surprinși. De fapt, ne aplaudam pe noi! Se poate spune că ne temeam de o surpriză neplăcută. Totuși, am fost optimiști până în momentul anunțării câștigătorilor.

Știu că toți ați participat în timpul liceului la diferite concursuri naționale și internaționale de informatică. Care au fost cele mai importante rezultate pe care le-ați obținut în această perioadă?

Mugurel-Ionuț Andreica: Cel mai important succes al meu este reprezentat de medalia de argint de la ediția 2001 a *Olimpiadei Internaționale de Informatică*. Tot în acest an, am obținut premiul I la *Olimpiada Națională de Informatică*; în anii precedenți a trebuit să mă mulțumesc cu premiile II și III. În anul 2000 m-am clasat pe primul loc la concursul internațional de programare desfășurat în *Rusia*, la *Iakutsk* și pe locul al treilea la concursul național de programare *Grigore Moisil* de la *Lugoj*. În acest an m-am clasat pe primul loc la trei concursuri importante: *Cupa Compaq*, *Concursul de Programare Agora* și *USACO Spring Contest*.

Florin Ghețu: Eu am obținut o medalie de aur la ediția de anul trecut a *Olimpiadei Balcanice de Informatică* și două medalii de argint la ultimele ediții ale *Olimpiadei de Informatică a Europei Centrale*. La *Olimpiada Națională de Informatică* am obținut premiul I în anul 2000, premiul II în 1998 și 1999 și premiul III în 2001. Am fost membru al lotului național de informatică în ultimii doi ani. La ediția din anul 1998 a concursului *Marele Premiu* PACO m-am clasat pe locul al doilea. În același an, m-am clasat pe aceeași poziție la concursul național de software *Grigore Moisil* de la *Lugoj*. La acest ultim concurs, am obținut premiul III în anul 1999. În anul 2000 m-am clasat pe primul loc la concursul interjudețean *Cristian S. Claude* de la *Galați*.

Andrei Gheorghe: La toate cele patru ediții ale Olimpiadei Naționale de Informatică la care am participat, am obținut premiul II. În 1999 și 2001

am fost membru al lotului național de informatică și, anul acesta, am obținut premiul I la concursul *Marele Premiu* PACO.

Care sunt, în opinia voastră, principalele diferențe dintre aceste concursuri și ACM?

Vom începe cu diferența evidentă: toate aceste concursuri au fost individuale, iar *ACM* este un concurs pe echipe. În plus, la *ACM* avem dreptul la mai multe încercări pentru o problemă, în timp ce, la celelalte concursuri, se evaluau o singură dată, variantele finale ale soluțiilor. Să nu uităm, însă, că la concursurile la care am participat în timpul liceului puteam obține punctaje parțiale la o problemă chiar dacă, din neatenție sau pentru că uitam anumite cazuri particulare, rezolvarea nu era perfectă. La *ACM* un program trebuie să furnizeze soluții corecte pentru toate seturile de date folosite la evaluare. O altă diferență este faptul că *ACM* tinde să fie un concurs puțin mai "tehnic" decât o olimpiadă de liceu. Totuși, în esență, diferențele nu sunt foarte mari; sunt necesare, în primul rând, o gândire algoritmică foarte bună și o viteză mare de implementare.

Ați fost reprezentanții universității care a organizat concursul. A constituit acest lucru un avantaj?

Din punctul nostru de vedere, nu. Este, totuși, un avantaj pentru universitate faptul că echipa ei a ieșit învingătoare.

În majoritatea universităților din România nu se acordă nici o importanță concursului ACM. Studenții afirmă că nimeni nu se ocupă de pregătirea lor. Este diferită situația în cazul vostru?

Și da, și nu. Până înaintea concursului au fost organizate baraje și concursuri de pregatire, care ne-au ajutat destul de mult. Credem că se vor mai organiza câteva până la *Finală*. Oricum, noi ne vom pregăti și singuri pentru a fi în formă maximă. Avem oricând la dispoziție *Internet*-ul pentru a găsi probleme interesante și... a nu sta degeaba.

Care sunt principalele modalități de pregătire pentru ACM?

O modalitate foarte bună de pregătire o constituie concursurile *online*, asemănătoare cu *ACM*, pe *site*-uri precum **acm.uva.es** sau **acm.timus.ru**. În rest, folosim aceleași tehnici pe care le foloseam, individual, pentru concursurile din timpul liceului. Fiecare caută probleme interesante sau pe care nu știe să le rezolve și le discută cu ceilalți. Comunicarea dintre noi este foarte importantă.

Credeți că experiența acumulată în timpul liceului a reprezentat un avantaj?

Bineînțeles, faptul că ne-am pregătit pentru concursurile naționale și internaționale la care am participat ne-a ajutat să acumulăm experiența pe care am folosit-o din plin și la *ACM*.

Credeți că GInfo poate contribui, în vreun fel, la pregătirea elevilor și studenților pentru concursurile de informatică?

Nu avem nici o îndoială. Dificultatea problemelor de la concursul de programare organizat de *GInfo* este destul de ridicată, ceea ce îi stimulează pe tinerii informaticieni. De asemenea, prezentările unor algoritmi sau ale soluțiilor unor probleme interesante sunt foarte folositoare.

În cazul vostru a contribuit GInfo în vreun fel?

Am învățat multe din rezolvările problemelor publicate în *GInfo*. Revista a constituit atât o sursă de probleme, cât și o posibilitate de verificare a corectitudinii unor metode de rezolvare (prin concursul de programare).

Până la finală mai sunt câteva luni. În martie, în Hawaii, vă veți confrunta cu cele mai bune echipe din lume. Sunteți pregătiți?

Da. Suntem pregătiți să învățăm mai multe și să lucrăm mai mult ca până acum. Dorim să obținem un loc cât mai bun la marea *Finală*, dar și să ne întoarcem de acolo bronzați.

Cine credeți că a contribuit cel mai mult la pregătirea voastră?

Mugurel-Ionuț Andreica: M-am pregătit foarte mult singur. Totuși, trebuie să recunosc că am învățat și din experiența altora. I-aș aminti aici pe ***d-na prof. Rodica Pintea*** și pe profesorii mei de la liceu (în special ***dl. prof. Emil Dragomirescu***), care m-au ajutat foarte mult, exact atunci când aveam nevoie.

Florin Gheț: Trebuie să mulțumesc în special ***d-nei prof. Daniela Lica*** de la Colegiul Național *Ion Luca Caragiale* din *Ploiești* și ***d-lui prof. Emil Onea***, inspector general de specialitate la *Ministerul Educației și Cercetării*, care m-au ajutat foarte mult și pot spune că îmi sunt mult mai mult decât simpli profesori. Nu în ultimul rând vreau să mulțumesc familiei pentru susținerea pe care mi-au acordat-o încă din clasa a V-a, când am participat la prima olimpiadă de informatică.

Andrei Gheorghe: Eu și cu ***Mugurel*** am fost la multe pregătiri și cercuri de informatică, din clasa a X-a până în clasa a XII-a. Aș dori să-i mulțumesc lui ***Cătălin Frâncu***, care, în clasa a IX-a, mi-a deschis interesul pentru algoritmi. De asemenea, ***d-na prof. Rodica Pintea*** a contribuit foarte mult la pregătirea elevilor bucureșteni care participă la olimpiade.

Felicitări încă o dată pentru victorie și mult succes la faza finală. Doriți să transmiteți un mesaj cititorilor GInfo?

Să aibă multă ambiție și încredere și să nu renunțe niciodată, indiferent ce și-ar propune! Să nu neglijeze nici școala, dar să se dedice trup și suflet informaticii, pentru că este, cu adevărat, ***știința viitorului***!

Bursele AGORA

S-a incheiat prima etapă a concursului Bursele Agora. În cadrul acestui număr vă vom prezenta soluțiile problemelor de la această rundă, clasamentul etapei, precum și cele trei probleme ale rundei a patra.

Prima rundă

Problemele primei runde a concursului nostru au fost destul de dificile. Prima dintre ele, *Matrice*, a fost rezolvată corect doar de trei concurenți, în timp ce la problema *Warcraft* nici unul dintre participanți nu a obținut punctajul maxim. Problema *Dune* a fost relativ simplă, ea putând fi rezolvată cu ajutorul algoritmului lui *Lee*, un algoritm clasic de programare dinamică.

Este îmbucurător faptul că nu au mai apărut erori banale cum ar fi folosirea *unit*-ului CRT de către programatorii în *Pascal*, așteptarea reacțiilor de la tastatură etc. Aceste erori nu au dispărut complet, dar numărul celor care au pierdut puncte datorită lor este nesemnificativ.

Pentru a studia datele de test folosite în cadrul evaluării puteți accesa paginile *Web* dedicate celor trei probleme ale primei runde. Acestea sunt:

- **www.ginfo.ro/concurs/matrice.shtml;**
- **www.ginfo.ro/concurs/dune.shtml;**
- **www.ginfo.ro/concurs/warcraft.shtml.**

Pe aceste pagini veți putea găsi și prezentările soluțiilor oficiale ale problemelor, precum și legături spre fișierele sursă ale rezolvărilor date de autori. Vă reamintim faptul că toate fișierele disponibile pe *site*-ul **www.ginfo.ro** sunt proprietatea editurii *Agora Media* și nu pot fi folosite în scopuri comerciale fără acordul scris al acesteia.

De asemenea, puteți obține informații referitoare la evaluarea problemelor. Clasamentul primei etape este disponibil la adresa **www.ginfo.ro/concurs/runda1/top.shtml.**

Clasamentul general al concursului este disponibil la adresa **www.ginfo.ro/concurs/top.shtml**

Această pagină conține lista tuturor celor care au trimis soluții pentru această rundă. În dreptul numelor apar punctajele obținute la cele trei probleme. Aceste punctaje sunt, de fapt, legături spre paginile personalizate ale concurenților. Un *click* pe unul dintre punctaje va duce la apariția unei pagini care oferă informații detaliate despre evaluarea problemei și modul în care a fost obținut punctajul respectiv.

Runda a patra

Cele trei probleme ale rundei a patra sunt *Arbori*, *Atlantis* și *Celule*.

Prima dintre ele a fost propusă de ***Mihai Stroe***, student în anul IV la *Universitatea Politehnica* din București, fost olimpic internațional, câștigător al primei ediții a concursului nostru și clasat pe locul al doilea la ediția de anul trecut.

Cea de-a doua problemă a fost propusă de ***Mihai Scorțaru***, student în anul IV la *Universitatea Tehnică* din Cluj, redactor-șef adjunct al revistei noastre și organizator principal al concursului.

Ultima problemă a fost aleasă de ***d-na lect. univ. Clara Ionescu***, cadru didactic în cadrul *Universității Babeș-Bolyai* din Cluj și redactor-șef al *GInfo*. Problema a mai fost propusă spre rezolvare la câteva concursuri de programare desfășurate acum câțiva ani.

Rezolvările celor trei probleme pot fi trimise până la data de ***28 februarie 2002***. Rezultatele oficiale ale rundei a patra vor deveni publice pe data de ***15 martie 2002***.

Pentru a evita întârzierile, vă recomandăm să trimiteți soluțiile cu câteva ore sau zile înaintea expirării termenului.

La mulți ani!

Șerban Gîlvitu, Hunedoara - 5 I 1986
Paul-Andrei Tuțu, Arad - 9 I 1984
Mihai Stoicescu, Boldești-Scăeni - 10 I 1984
Ștefan Filip, Bacău - 12 I 1985
Nicolae Tanase-Nicola, Slatina - 13 I 1985
Constantin Asofiei, Târgu Neamț - 14 I 1984
Mugurel-Ionuț Andreica, București - 16 I 1983
Radu-Victor Sărmășag, Târgoviște - 17 I 1986
Romulus Ghiorghiaș, Brașov - 18 I 1980
Romulus-Nicolae Apolzan, Timișoara - 18 I 1982
Adrian-Gheorghe Ceaușu, Alba Iulia - 18 I 1986
Cristian Băicoianu, Ploiești - 20 I 1984
Andrei Gheorghe, București - 27 I 1983
Mircea-Șerban Savin, Iași - 30 I 1984
Andrei Ionescu, Arad - 31 I 1985

Runda 1, problema 1

MATRICE

Au rezolvat corect:

Adrian-Nicolae Cârcu, Bistrița
Radu Berinde, București
Csaba András, Oradea

Această problemă a fost rezolvată corect doar de către trei dintre cei care au participat la prima rundă a ediției 2001/2002 a concursului de programare *Bursele Agora*. Aceasta dovedește faptul că problema a fost dificilă, dar rezolvabilă.

Metoda folosită de către autor pentru rezolvarea problemei este ***programarea dinamică***.

Considerăm matricea de intrare X. Vom începe cu câteva observații ajutătoare. Pentru început, numărul mic al coloanelor ar trebui să ne pună pe gânduri. Apare în mod evident întrebarea dacă această limitare ne poate ajuta la rezolvarea problemei.

Se observă că există cel mult 64 de modalități de a plasa pătrate (mai exact colțurile din stânga-sus ale acestora) pe o linie a matricei. De ce 64? Pentru că pătratele trebuie să fie incluse complet în matrice, deci colțul din stânga-sus al unui pătrat poate fi plasat numai pe una dintre primele șase coloane. De fapt, acest număr poate fi redus la 37, deoarece unele posibilități de plasare sunt inutile.

De exemplu, se observă că patru pătrate sunt suficiente pentru a acoperi întreaga linie, deci nu are sens să plasăm cinci pătrate pe o linie, deoarece cel puțin unul dintre ele acoperă poziții acoperite și de alte pătrate, deci poate fi eliminat, suma elementelor acoperite rămânând aceeași. De asemenea, nu are rost să plasăm colțurile din stânga-sus a trei pătrate în trei coloane consecutive, deoarece pozițiile acoperite de pătratul din mijloc, sunt acoperite de celelalte două pătrate.

Presupunem că știm, pentru o anumită linie, sumele maxime care pot fi obținute plasând 0, 1, 2, ..., K pătrate pe primele i linii folosind pe linia i o anumită configurație. Apare întrebarea dacă putem determina aceleași informații pentru linia $i + 1$.

Răspunsul este afirmativ și ne oferă rezolvarea problemei. Construim (cel puțin teoretic) tabloul tridimensional `M[1..N-1,0..K,1..37]`. Un element `M[i,j,l]` reprezintă suma maximă obținută plasând `j` pătrate pe primele `i` linii folosind configurația `l` pe linia `i`.

Elementele matricei `M[i+1]` pot fi determinate folosind matricea `M[i]`. Pentru a calcula un element al matricei `M[i+1]` avem nevoie de următoarele informații:

- suma elementelor de pe linia `i+2` care sunt acoperite de pătratele care vor fi plasate pe linia `i+1`, folosind configurația `l`;
- numărul `nr[l]` al pătratelor care sunt folosite pentru obținerea configurației `l`;
- suma elementelor de pe linia `i+1` care sunt acoperite folosind configurația `l`, dar nu și folosind o configurație `c` (determinăm valorile sumei pentru orice configurație `c`);
- valoarea maximă din `M` pentru fiecare configurație `c` plasată pe linia `i` care conține `j-nr[l]` pătrate.

Modul în care determinăm aceste informații poate fi înțeles studiind fișierul sursă.

Programul este construit astfel:

- configurațiile posibile de pe o linie sunt păstrate în matricea de constante `p`, iar numărul de pătrate folosit pentru obținerea fiecărei configurații este păstrat în vectorul `nr`;
- tabloul tridimensional `M` este simulat cu ajutorul a două matrice `A` și `B` (adică `A=M[i]`, se calculează `B=M[i+1]`, apoi `A` devine `B` etc.)
- pentru fiecare rând unde se pot plasa pătrate
  - se calculează suma adăugată pe rândul respectiv dacă plasăm o anumită configurație, în funcție de configurația plasată pe rândul anterior;
  - se calculează suma adăugată pe rândul ***următor*** dacă plasăm o anumită configurație pe rândul curent;
  - se inițializează matricea `B`;
  - se realizează calculul valorilor din `B` (datorită faptului că operațiile din această zonă sunt executate de foarte multe ori, codul trebuie optimizat la maximum; programul prezentat aici a fost optimizat după obținerea unei versiuni funcționale, dar mai lente; sunt posibile și alte optimizări, dar am preferat să ne limităm la acestea pentru a nu afecta lizibilitatea codului; versiunea pe care v-o punem la dispoziție se încadrează în limita de o secundă pentru orice set de date dacă se folosește cu o configurație echivalentă celei specificate în regulamentul concursului *Bursele Agora*);
  - `A` devine `B`.

Soluția este dată de elementul maxim obținut în final în matricea `A`.

Codul sursă al programului este disponibil pentru *download* la adresa **www.ginfo.ro/concurs/matrice.shtml**.

Runda 1, problema 2

DUNE

Această problemă a fost rezolvată corect de 49 dintre participanții la prima rundă a concursului *Bursele Agora*, ceea ce demonstrează faptul că rezolvarea sa nu ridică dificultăți deosebite. Problema poate fi rezolvată foarte ușor folosind algoritmul lui *Lee*. La fiecare pas vom memora toate pozițiile în care poate ajunge *Muad'Dib*.

Inițial *Muad'Dib* se află în poziția marcată cu `'*'`. Vom marca cu `'#'` toate regiunile în care se poate ajunge din poziția inițială mergând în direcția indicată de primul element al șirului de caractere dat.

La fiecare dintre următorii pași vom efectua următoarele operații:

- vom marca cu `'.'` toate pozițiile care nu sunt marcate cu `'+'` sau cu `'#'`;
- vom marca cu `'*'` toate pozițiile care sunt marcate cu `'#'`;
- vom marca cu `'+'` poziția în care se află *sietch*-ul de plecare:
- pentru fiecare poziție marcată cu `'*'`:
  - ♦ vom marca cu `'#'` toate regiunile în care se poate ajunge din poziția considerată mergând în direcția indicată de elementul curent al șirului de caractere dat;
  - ♦ există posibilitatea să marcăm cu `'#'` poziții care au fost marcate cu `'*'`; acest lucru nu afectează corectitudinea algoritmului, deoarece din pozițiile care au fost marcate cu `'*'` nu se poate ajunge în alte poziții decât cele deja marcate cu `'#'`.

În final, soluția va fi dată de numărul pozițiilor marcate cu `'#'`.

Aceasta este o prezentare teoretică a rezolvării. Practic, pentru simplitate, nu se mai face diferență între primul și ceilalți pași ai algoritmului. Astfel, se iau în considerare toate pozițiile marcate cu `'*'`, se marchează cu `'#'` toate pozițiile în care se poate ajunge, se marchează cu `'.'` toate pozițiile marcate cu `'*'`, se marchează cu `'+'` poziția *sietch*-ului de pornire și se repetă acești pași până când nu mai are loc nici o schimbare de direcție. În final, se numără pozițiile marcate cu `'*'`.

Mulți dintre concurenți nu au luat în considerare anunțul general care indica faptul că *sietch*-ul de pornire reprezintă o regiune stâncoasă deoarece s-au înscris după publicarea acestuia și nu au mai primit mesajul în care se preciza că acest anunț poate fi citit pe *site*-ul oficial al concursului. Am decis ca rezolvările care nu iau în considerare acest anunț să fie considerate corecte.

Codul sursă al programului este disponibil pentru *download* la adresa **www.ginfo.ro/concurs/dune.shtml**.

Au rezolvat corect:

Cristian Alexandrescu, Botoșani
Csaba András, Oradea
Mugurel-Ionuț Andreica, București
Liviu-Cosmin Andreicuț, Bacău
Adrian Balașko, Zalău
Andrei Benea, Focșani
Adrian-Nicolae Cârcu, Bistrița
Mihai-Daniel Cilidariu, Botoșani
Phuong Dao The, Vietnam
Vlad Dascălu, Bacău
Lorena-Iulia Dăian, Cluj-Napoca
Tiberiu Dănet, București
Radu Dondera, București
George-Adrian Drumea, București
Dorin-Daniel Dudău, Târgu-Jiu
Octavian-Daniel Dumitran, București
Ștefăniță Fechete, Suceava
Ștefan Gheorghe, București
Andrei Giurgiu, București
Ioan-Claudiu Gruia, București
Andrei Homescu, Târgu-Jiu
Liviu-Laurențiu Iacob, București
Alexandru Ivan, București
Slotta Johannes, Germania
Andrew Kirilenko, Belarus
Cosmin Lehene, Zalău
Maximilian Machedon, București
Aurelian Marinescu, Brașov
Andrei Markovits, Satu Mare
Marius-Dorin Moraru, Buftea
Alexandru Moșoi, Bacău
Silvestru-Cosmin Negruțeri, Bistrița
Bogdan Nicolae, Sibiu
Alexandru Oprea, Sighișoara
Sorin Otescu, Brașov
Alexandru Palade, Târgu Mureș
Mihai Pantelimon, București
Csaba Pătcaș, Oradea
Alpár-Ferenc Perini, Sfântu Gheorghe
Alexandru Radu, Bârlad
Claudiu Raicu, Pitești
Constantin Răuțu, Botoșani
Lucian-Iulian Saucă, Brașov
Dan-Octavian Savu, București
Radu Andrei Ștefan, Brașov
Eszter Tasi, Sfântu Gheorghe
Silviu-Gabriel Udrea, Târgu-Jiu
Andrei Vancea, Cluj-Napoca
Mihai Voinescu, Târgu-Jiu

Runda 1, problema 3

WARCRAFT

Nici unul dintre participanți nu a reușit să obțină toate cele 100 de puncte corepunzătoare acestei probleme.

Cele mai mari punctaje au fost obținute de ***Mugurel-Ionuț Andreica*** din *București* (62 de puncte), ***Constantin Asofiei*** din *Târgu Neamț* (53 de puncte) și ***Ștefăniță Fechete*** din *Suceava* (50 de puncte).

Dintre cei 86 de concurenți care au trimis o soluție pentru această problemă, doar nouă au reușit să obțină cel puțin 20 de puncte.

Până în acest moment nu a fost găsit un algoritm polinomial pentru rezolvarea acestei probleme, deci o putem considera ca fiind *NP*-completă chiar dacă nu avem la dispoziție o demonstrație matematică a acestui fapt. Lăsând la o parte metoda *backtracking*, rămâne să căutăm o rezolvare bazată pe metode euristice sau cu ajutorul unui algoritm probabilistic.

În cele ce urmează vom prezenta un algoritm evolutiv care poate fi folosit pentru rezolvarea acestei probleme.

Reamintim că algoritmii evolutivi constau în alegerea unei populații inițiale și apoi efectuarea unor mutații asupra acestora cu scopul de a îmbunătăți soluțiile obținute. În final, este aleasă cea mai bună soluție obținută. Evident, nu se poate garanta că această soluție este întotdeuana cea corectă, dar practica arată că, în majoritatea situațiilor, soluția descrisă este acceptabilă.

Mai întâi va trebui să alegem structura unui membru al populației. Alegerea nu este foarte dificilă, un membru poate fi reprezentat de o matrice cu elemente `'+'` și `'.'`, unde fiecare simbol reprezintă teritoriile uneia dintre națiuni.

O soluție va reprezenta configurația hărții după efectuarea schimburilor de teritorii. Evident, numărul elementelor cu valoarea `'+'` trebuie să fie identic cu numărul elementelor cu valoarea `'+'` citite la intrare. Analog, numărul elementelor cu valoarea `'.'` trebuie să fie identic cu numărul elementelor cu valoarea `'.'` citite la intrare.

De asemenea, trebuie să existe doar două zone disjuncte compacte: una ale cărei elemente au valoarea `'+'` și una ale cărei elemente au valoarea `'.'`.

Trebuie să găsim o măsură a performanței unui individ. Aceasta poate fi găsită foarte ușor: cu cât există mai puține diferențe între configurația individului și harta inițială, cu atât individul este mai performant. Evident, indivizii pentru care există mai mult de două zone disjuncte, trebuie penalizați. Vom aplica câte o penalizare pentru fiecare zonă suplimentară găsită.

Conceptul de algoritm evolutiv implică folosirea unui singur operator pentru îmbunătățirea soluțiilor și anume ***mutația***.

Prin mutație se efectuează o mică modificare asupra configurației corespunzătoare unui individ. În cazul nostru cea mai mică modificare este reprezentată de un schimb de teritorii. Pentru aceasta vom alege aleator o poziție căreia îi corespunde un element cu valoarea `'+'` și o poziție căreia îi corespunde un element cu valoarea `'.'` și vom interschimba valorile celor două elemente. După efectuarea mutației vom studia performanța noului individ și în cazul în care elementul este mai performant decât elementul asupra căruia se efectuează mutația, noul elemenet este inclus în populație, iar vechiul element este eliminat.

În final vom alege cel mai performant individ care, în majoritatea cazurilor, reprezintă soluția problemei. Vom număra pozițiile în care elementul din configurația corespunzătoare individului diferă de elementul din configurația inițială (citită din fișierul de intrare). Numărul schimburilor de teritorii este obținut prin înjumătățirea acestei valori și va fi scris în fișierul de ieșire. Înjumătățirea se datorează faptului că printr-un schimb de teritorii apar diferențe în două poziții, deci numărul diferențelor este egal cu dublul schimburilor de teritorii.

În cazul acestei probleme, un rol foarte important îl are generarea populației inițiale. Algoritmul standard impune generarea unor populații aleatoare, dar această regulă nu se aplică în acest caz.

Este indicat ca, pentru generarea populației inițiale, să folosiți câteva metode euristice de găsire a soluției. În unele cazuri aceste metode vor duce chiar la găsirea soluției dar, în majoritatea cazurilor vor duce doar la o configurație mai mult sau mai puțin apropiată de soluție. Totuși, numărul mutațiilor care trebuie efectuate asupra configurațiilor obținute prin metode euristice este mult mai mic și există șanse mult mai mari de obținere a soluțiilor.

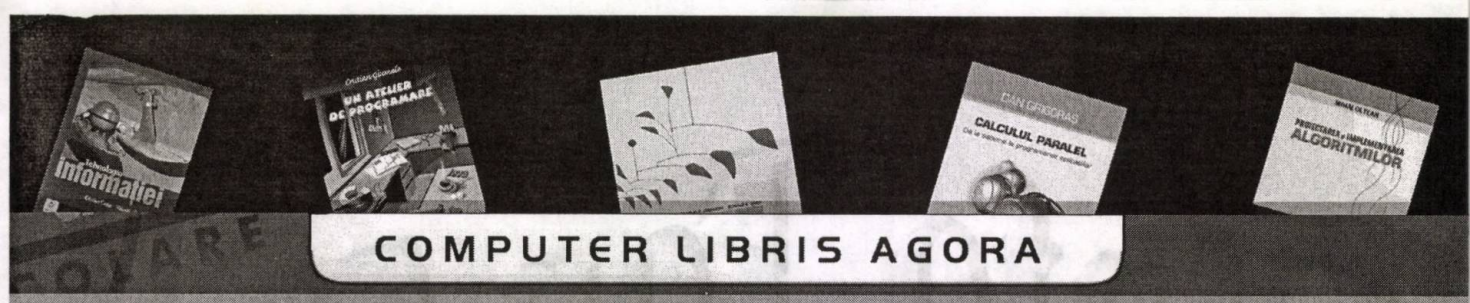
Datorită faptului că nici un concurent nu a rezolvat integral problema, am decis să nu prezentăm metodele euristice pe care le-am folosit pentru generarea populației inițiale.

Va trebui să vă folosiți imaginația pentru a găsi astfel de metode care să ducă la obținerea de soluții cât mai apropiate de cea corectă.

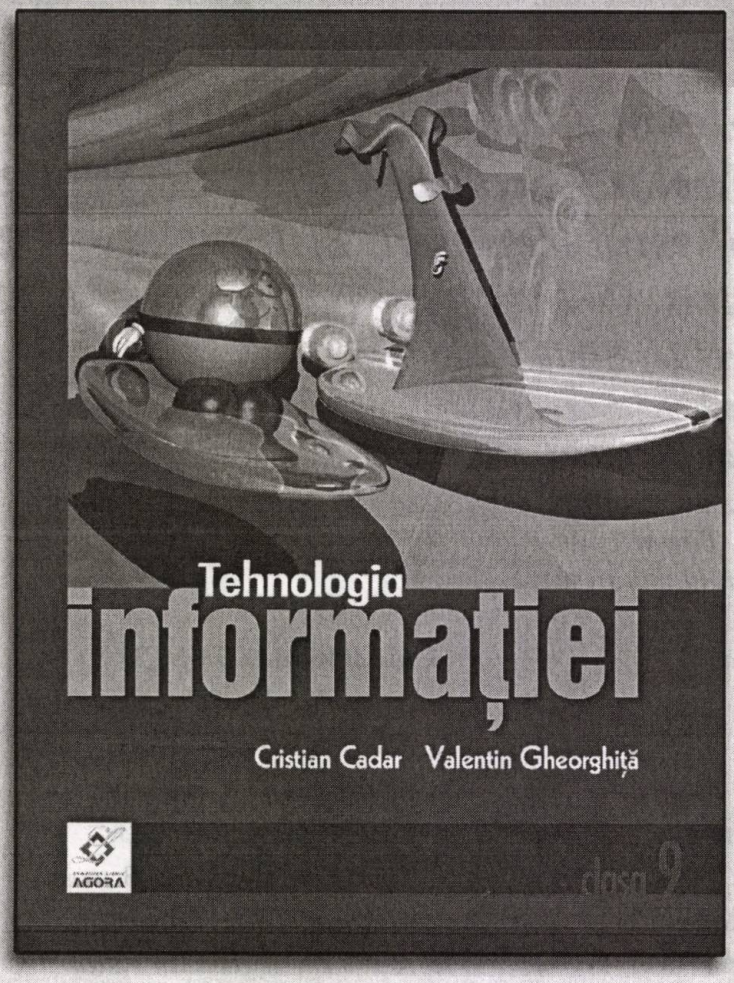
Schema codului sursă al programului este disponibilă pentru *download* la **www.ginfo.ro/concurs/warcraft.shtml**.

GInfo Top 50

1.	Mugurel-Ionuț Andreica, București	251
2.	Adrian-Nicolae Cârcu, Bistrița	221
3.	Cosmin-Silvestru Negrușeri, Bistrița	218
4.	Csaba András, Oradea	213
5.	Bogdan Nicolae, Sibiu	192
6.	Ștefăniță Fechete, Suceava	187
7.	Tiberiu Dăneț, București	183
8.	Radu Berinde, București	181
9.	Claudiu Raicu, Pitești	169
10.	Ioan-Claudiu Gruia, București	166
11.	Andrei Vancea, Cluj-Napoca	164
12.	Adrian Balașko, Zalău	157
13.	Csaba Pătcaș, Oradea	155
14.	Octavian-Daniel Dumitran, București	148
14.	Liviu-Laurențiu Iacob, București	148
16.	Andrew Kirilenko, Belarus	145
16.	Alexandru Palade, Târgu Mureș	145
18.	Marius-Dorin Moraru, Buftea	144
19.	Valentin Gușă, București	141
20.	Andrei Benea, Focșani	140
20.	Cosmin Lehene, Zalău	140
20.	Aurelian Marinescu, Brașov	140
20.	Mihai Pantelimon, București	140
24.	George-Adrian Drumea, București	139
25.	Liviu-Cosmin Andreicuț, Bacău	137
25.	Vadim Barkalov, Statele Unite	137
25.	Ștefan Gheorghe, București	137
25.	Andrei Homescu, Târgu-Jiu	137
25.	Alpár-Ferenc Perini, Sfântu Gheorghe	137
25.	Eszter Tasi, Sfântu Gheorghe	137
25.	Mihai Voinescu, Târgu-Jiu	137
32.	Maximilian Machedon, București	135
33.	Andrei Giurgiu, București	134
34.	Alexandru Oprea, Sighișoara	132
35.	Radu-Andrei Ștefan, Brașov	130
36.	Mihai-Daniel Cilidariu, Botoșani	129
37.	Constantin Răuțu, Botoșani	126
38.	Andrei David, Râmnicu Vâlcea	125
39.	Vlad Dascălu, Bacău	122
40.	Lorena-Iulia Dăian, Cluj-Napoca	116
40.	Tiberiu Boroș, Pitești	116
42.	Ion Ionașcu, București	115
42.	Sorin Suciu, Craiova	115
44.	Cristian Alexandrescu, Botoșani	114
44.	Dan-Octavian Savu, București	114
46.	Phuong Dao The, Vietnam	108
46.	Alexandru Ivan, București	108
46.	Sorin Otescu, Brașov	108
46.	Alexandru Radu, Bârlad	108
50.	Dorin-Daniel Dudău, Târgu-Jiu	106



COMPUTER LIBRIS AGORA



Lucrarea de față își propune să pregătească cititorul în scopul utilizării unui calculator personal. Ea urmărește programa școlară, aprobată de Ministerul Educației Naționale, pentru disciplina "Tehnologia informației", disciplină în planul de învățământ a claselor a IX-a la profilul teoretic și pedagogic. Manualul corespunde și programei de la disciplina "Informatică - Tehnologii asistate pe calculator", profilul tehnologic.

Pentru parcurgerea acestei cărți nu sunt necesare cunoștințe prealabile în domeniu, termenii și noțiunile prezentate sunt explicate detaliat în carte. Exemplele sunt sugestive, cartea nu este încărcată inutil cu informații care depășesc programa amintită.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

Runda 4, problema 1

ARBORI

Denumirea subdirectorului: **TREES**

Denumirea fișierului de intrare: **TREES.IN**

Denumirea fișierului de ieșire: **TREES.OUT**

Descrierea problemei

În curtea unei case se află M arbori binari. Într-o zi, proprietarul a primit de la un prieten o fotografie cu N arbori binari. Poza i-a plăcut atât de mult, încât a decis să transforme N dintre arborii din grădină în cei N arbori din poză, dacă este posibil. Dacă nu, el dorește să transforme un număr K dintre arborii din grădină în K arbori (cu numere de ordine diferite) din poză, maximizând valoarea K.

Proprietarul poate tăia crengi ale arborilor din grădină. În final, doi arbori (unul din gradină, unul din poză) sunt identici dacă au aceeași configurație a crengilor (muchiilor); numerele care identifică nodurile sunt ignorate. Mai multe detalii reies din studierea exemplului.

Trebuie să scrieți un program pentru a-l ajuta pe proprietar să minimizeze numărul crengilor tăiate, în condițiile în care numărul K, definit anterior, este maxim.

Date de intrare

Pe prima linie a fișierului de intrare se află valorile M și N, separate printr-un singur spațiu.

În continuare sunt descriși cei M arbori din curte. Pe prima linie corespunzătoare unui arbore se află numărul X al nodurilor arborelui. Fiecare dintre următoarele linii descriu fiii fiecărui nod din arbore. Prima linie corespunde primului nod (care este întotdeauna rădăcina arborelui), a doua celui de-al doilea nod etc. Fiecare linie conține două numere cuprinse între 0 și X și indică numărul de ordine al celor doi fii ai nodului corespunzător liniei. Valoarea 0 indică faptul că nodul respectiv nu există.

Urmează descrierile celor N arbori din poză. Aceștia sunt descriși în același mod ca și cei M arbori din curte.

Datele de intrare sunt corecte; se garantează că fiecare descriere corespunde unui arbore binar.

Date de ieșire

Pe prima linie a fișierului de ieșire se va afla numărul K al arborilor care pot fi obținuți. Pe a doua linie se află numărul MIN al crengilor tăiate.

Pe fiecare dintre următoarele K linii se va afla o pereche de numere: *număr_arbore_curte*, *număr_arbore_poză* care indică faptul că arborele *număr_arbore_poză* se obține tăind crengi din arborele *număr_arbore_curte*. Numerele de ordine ale arborilor din curte care apar în fișierul de ieșire trebuie să fie distincte. Similar, numerele de ordine ale arborilor din poză care apar în fișierul de ieșire trebuie să fie distincte. Nu contează ordinea în care sunt scrise perechile în fișierul de ieșire, iar dacă există mai multe soluții se va afișa una singură.

Restricții

- $1 \le M, N \le 40$;
- doi arbori nu se consideră identici decât dacă subarborii lor drepți sunt identici și subarborii lor stângi sunt identici; dacă cei doi arbori sunt simetrici (subarborele stâng al unuia este identic cu subarborele drept al celuilalt și invers) atunci ei nu sunt considerați identici;
- fiecare arbore are cel mult 100 de noduri.

Exemplu

TREES.IN

```
3 3
2
2 0
0 0
3
2 3
0 0
0 0
5
5 3
0 0
0 0
0 0
4 2
```

(continuare)

```
3
3 2
0 0
0 0
2
0 2
0 0
3
2 0
0 3
0 0
```

TREES.OUT

```
2
1
2 1
3 2
```

Timp de execuție/test: 1 secundă

Runda 4, problema 2

ATLANTIS

Denumirea subdirectorului: **ATLANTIS**
Denumirea fișierului de intrare: **ATLANTIS.IN**
Denumirea fișierului de ieșire: **ATLANTIS.OUT**

Descrierea problemei

În *Anul Domnului Două Mii Patru Sute Cincizeci și Șapte* au fost, în sfârșit, descoperite vechile ruine ale civilizației *Atlantidei.*

Expediția a fost condusă de marele explorator al oceanelor *Iahim Uratrox*. Acesta a ajuns în fața porții miticului oraș *Atlantis*, dar nu a reușit să o deschidă deoarece este protejată de un vechi cifru.

Pe poartă sunt inscripționate zece simboluri distincte. Oamenii de știință au descoperit că acestea reprezintă cele 10 cifre zecimale, prima dintre ele corespunde cifrei `0`, a doua corespunde cifre `1` și așa mai departe.

Pe mecanismul cifrului sunt reprezentate patru dintre cele zece simboluri. Există posibilitatea ca două sau mai multe dintre cele patru simboluri să fie identice. Mecanismul cifrului constă din N dispozitive care sunt activate prin simpla atingere. Pe fiecare dintre aceste N dispozitive este gravată o inscripție corespunzătoare unei cifre.

Oamenii de știință au mai descoperit că cifrul este dat de cel mai mic multiplu strict pozitiv al numărului format din cele patru simboluri desenate pe mecanism. Datorită faptului că pentru formarea multiplului nu sunt disponibile decât cifrele corespunzătoare celor N dispozitive, doar acestea pot fi folosite pentru formarea multiplului.

Operația de deschidere a porții este acum foarte simplă. Trebuie activate, succesiv, dispozitivele corespunzătoare cifrelor care formează multiplul. După activarea unui dispozitiv se aprinde o lumină care indică faptul că cifra a fost acceptată, iar apoi lumina se stinge pentru a permite repetarea cifrei.

Sarcina voastră este de a-l ajuta pe conducătorul expediției să găsească cifrul. Veți avea la dispoziție cele patru cifre ale numărului de la care se pornește și cifrele pe care le aveți la dispoziție pentru a determina multiplul.

Date de intrare

Pe prima linie a fișierului de intrare se află cele patru cifre ale numărului gravat pe mecanismul cifrului, neseparate prin spații. Există posibilitatea ca prima, primele două sau primele trei cifre să fie `0`. Totuși, cel puțin una dintre cele patru cifre este diferită de zero.

Pe a doua linie a fișierului se află numărul N al cifrelor pe care le aveți la dispoziție pentru a forma multiplul.

Următoarele N linii conțin, fiecare, câte o cifră pe care o aveți la dispoziție.

Date de ieșire

Fișierul de ieșire va conține o singură linie pe care se vor afla cifrele multiplului neseparate prin spații.

În cazul în care un astfel de multiplu nu poate fi determinat, *Iahim Uratrox* va folosi o încărcătură explozivă pentru a deschide poarta, iar dumneavoastră veți scrie în fișierul de ieșire valoarea `0`.

Restricții și precizări

- $1 \le N \le 10$;
- cele N cifre pe care le aveți la dispoziție sunt distincte;
- valoarea numărului inscripționat pe mecanismul cifrului este cuprinsă între 1 și 9999;
- în cazul în care aveți la dispoziție cifra `0`, aceasta nu poate fi prima cifră a multiplului pe care îl veți afișa.

Exemplu

ATLANTIS.IN

```
0012
5
5
4
1
7
9
```

ATLANTIS.OUT

```
144
```

Timp de execuție/test: **1 secundă**

Runda 4, problema 3

CELULE

Denumirea subdirectorului: **CELLS**

Denumirea fișierului de intrare: **CELLS.IN**

Denumirea fișierului de ieșire: **CELLS.OUT**

Descrierea problemei

Un număr de n celule sensibile sunt așezate în cerc, fiecare comunicând cu cei doi vecini ai săi. O celulă se poate găsi în două stări: *excitată* sau *liniștită*. Dacă o celulă este excitată la un moment dat t, atunci ea emite un semnal care ajunge la cele două celule vecine la momentul $t + 1$. Fiecare celulă este excitată atunci și numai atunci, când la ea ajunge un semnal de la una din celulele vecine. Dacă semnalele ajung deodată din ambele părți, atunci ele se anulează.

Dându-se o configurație inițială de celule excitate și liniștite, se cere să se determine dacă excitația se menține oricât de mult, sau se va liniști.

Date de intrare

Prima linie a fișierului de intrare conține un număr natural m, reprezentând numărul configurațiilor din fișier.

Fiecare din configurații se precizează pe două linii: pe prima linie se află un număr natural n, reprezentând numărul celulelor din configurație, iar pe a doua se află configurația propriu-zisă.

O configurație este reprezentată printr-o succesiune de cifre (nedespărțite prin spații) din mulțimea {0, 1}, unde 1 indică faptul că celula este excitată, iar 0 indică faptul că celula este liniștită.

Date de ieșire

Fișierul de ieșire va conține câte o linie pentru fiecare configurație descrisă în fișierul de intrare. În cazul în care excitația se menține oricât de mult, pe linia corespunzătoare se va scrie șirul de caractere YES. În caz contrar, linia va conține șirul de caractere NO.

Restricții și precizări

- $1 \le m \le 30$;
- $1 \le n \le 10000$;
- nu se va acorda punctajul corespunzător unui test decât dacă toate cele m răspunsuri sunt corecte; nu se acordă punctaje parțiale.

Exemplu

CELLS.IN	CELLS.OUT
3	NO
12	YES
111111111111	NO
3	
110	
6	
101010	

Explicație

Pentru șirul format din 12 celule (toate fiind excitate), excitația nu se menține decât în primul moment de timp. La al doilea moment, fiecare celulă va primi câte un semnal de la cele două celule vecine, deci aceste semnale se vor anula reciproc. Ca urmare, nici una dintre celule nu va mai fi excitată la al doilea moment de timp, deci excitația nu se va menține.

Pentru șirul format din 3 celule, dintre care primele două sunt excitate, configurația nu se modifică, deci excitația se va menține oricât de mult. Prima celulă va rămâne excitată deoarece va primi semnal doar de la a doua celulă; a doua celulă va rămâne și ea excitată deoarece va primi semnal doar de la prima celulă; a treia celulă va rămâne liniștită pentru că primește semnal atât de la prima, cât și de la a doua celulă.

Pentru șirul format din 6 celule dintre care cele de pe pozițiile impare sunt excitate, iar cele de pe pozițiile pare sunt liniștite, excitația se menține, din nou, doar la primul moment de timp. La al doilea moment de timp celulele de pe pozițiile impare nu vor primi nici un semnal de la celulele vecine, iar celule de pe pozițiile pare vor primi câte un semnal de la cele două celule vecine, semnale care se vor anula reciproc. Așadar, în acest moment, nici una dintre celule nu este excitată, deci excitația nu se va menține, ci se va liniști.

Timp de execuție/test: 1 secundă

Metode de SORTARE

Mihai Scorțaru

Operația de ordonare a unor articole în funcție de diverse criterii este foarte des întâlnită în practică. Se cunosc o mulțime de algoritmi de sortare, majoritatea fiind foarte simpli. În cadrul acestui articol vom încerca să realizăm o prezentare comparativă a performanțelor care pot fi obținute folosind două dintre cele mai cunoscute metode de sortare.

Introducere

Metodele de sortare cele mai des folosite pot fi clasificate în două categorii: *metode directe* și *metode avansate*.

Metodele directe se bazează pe algoritmi de dificultate redusă, ușor de găsit și de înțeles. Metodele directe pe care le vom lua în considerare sunt sortarea prin selecție (*SelectSort*), sortarea prin inserție (*InsertSort*) și sortarea cu bule (*BubbleSort*).

Metodele avansate se bazează pe algoritmi puțin mai complicați, dar care nu necesită cunoștințe avansate de algoritmică. Câteva din cele mai cunoscute sunt sortarea rapidă (*QuickSort*), sortarea prin interclasare (*MergeSort*) și sortarea cu ansamble (*HeapSort*).

Vom studia doar performanțele a două metode directe de sortare și anume sortarea prin selecție și sortarea prin inserție. Este cunoscut faptul că nu există algoritmi generali de sortare care să aibă, în cel mai defavorabil caz, un ordin de complexitate mai mic decât $\Omega(n \cdot \log n)$. Vom studia ordinele de complexitate ale algoritmilor prezentați pentru cazul cel mai favorabil, cazul mediu și cazul cel mai defavorabil. După cum veți vedea, în cazul unor anumite metode este foarte dificil sau imposibil de determinat cazul cel mai defavorabil.

Măsurarea performanțelor

Vom studia performanțele algoritmilor din punct de vedere teoretic. Pentru aceasta va trebui să alegem o măsură a performanțelor acestora.

Nu vom putea compara doi algoritmi doar pe baza ordinului de complexitate deoarece, în multe cazuri, cei doi algoritmi vor avea același ordin de complexitate.

Nu putem măsura nici timpii de execuție ai unor programe, deoarece vom prezenta doar teoretic algoritmii, fără a studia efectiv și implementările.

Pentru a determina măsura performanțelor vom identifica operațiile care sunt efectuate în timpul execuției.

Trebuie remarcat faptul că, de obicei, nu se sortează șiruri de numere, ci șiruri de articole, deci o comparare între două articole implică, în general, mult mai multe operații decât o comparare între două numere întregi.

Este evident că vom efectua parcurgeri (liniare sau nu) ale șirurilor care trebuie sortate. Așadar, vom avea nevoie de anumiți indecși care vor fi folosiți pentru aceste parcurgeri. Acești indecși sunt, de obicei, numere întregi și operațiile efectuate asupra lor sunt rapide.

Pentru a sorta elementele șirului vom avea nevoie de comparații. O comparare între două elemente ale vectorului este o operație mult mai lentă.

De asemenea, vom efectua diferite atribuiri. Durata unei operații de atribuire este comparabilă cu cea a unei operații de comparare, dar nu se poate spune că cele două durate sunt egale.

Așadar, vom lua în considerare doar operațiile de comparare și atribuire care sunt efectuate asupra elementelor vectorului sau asupra unor elemente care au aceeași structură cu elementele vectorului. Cel mai bun exemplu de astfel de elemente îl reprezintă variabilele auxiliare.

Nu vom lua în considerare operațiile de comparare sau atribuire efectuate asupra indecșilor sau a altor elemente având structură similară.

De fapt, practic, atribuirile pot fi evitate în totalitate dacă operațiile nu se efectuează asupra elementelor vectorului propriu-zis, ci asupra unui vector auxiliar de indecși. Deși, în practică, această metodă este folosită aproape întotdeauna, vom lucra direct cu elementele șirului pentru a obține o mai bună caracterizare a performanțelor algoritmilor studiați.

Din aceste motive, trebuie reținut faptul că o măsură a performanțelor general acceptată este reprezentată de ***numărul comparărilor efectuate***.

Totuși, în cele ce urmează, vom folosi două măsuri diferite ale performanței pentru a caracteriza algoritmii și

anume: ***numărul comparărilor efectuate*** și ***numărul atribuirilor efectuate.***

Apare în mod evident întrebarea: *cum vom determina numărul comparărilor și numărul atribuirilor*? Răspunsul este unul foarte simplu: *le vom număra.*

Pentru a număra comparările și atribuirile vom folosi două variabile care vor fi incrementate de fiecare dată când se execută o comparare de articole sau o atribuire de articole. În acest articol vom denumi aceste variabile `comp` și `atr`.

În cazul în care o comparare apare în cadrul condiției de continuare a unei structuri repetitive anterior condiționate, atunci se va executa o comparare pentru fiecare execuție a corpului buclei și una suplimentară în momentul în care condiția de executare a corpului buclei nu mai este satisfăcută.

În cazul în care compararea apare în cadrul condiției de continuare a unei structuri repetitive posterior condiționate, nu se execută și compararea suplimentară, fiind executate doar comparările corespunzătoare execuțiilor corpului buclei.

În cazul în care compararea apare în cadrul condiției decizionale a unei structuri alternative, numărul comparațiilor va crește indiferent de rezultatul evaluării condiției.

Vom determina numărul de atribuiri și comparații pentru șiruri formate din 10, 20, 30, ..., 100 de numere.

În continuare vom prezenta metodele directe de sortare și vom realiza o prezentare comparativă a performanțelor acestora.

Pentru simplificare, vom considera că vom sorta un șir de n numere întregi și că acestea trebuie ordonate crescător. Această alegere nu reduce generalitatea, deoarece vom lua în considerare doar comparațiile și atribuirile efectuate asupra elementelor vectorului și asupra variabilelor auxiliare. Vom ignora toate celelalte atribuiri și comparări, chiar dacă ele sunt efectuate tot asupra unor numere întregi.

Sortarea prin selecție

Ideea care stă la baza sortării prin selecție este aceea de a alege primul element al șirului sortat și a-l așeza pe prima poziție. Presupunem că acest element s-a aflat pe poziția x în șirul inițial; primul element în șirul inițial va fi mutat în poziția x. În continuare vom aplica aceeași regulă pentru restul șirului (începând cu al doilea element) și vom obține primele două elemente ale șirului sortat. Vom aplica apoi regula pentru al treilea element, al patrulea element etc. până când vom obține șirul sortat.

Prezentarea algoritmului

Se observă că vom parcurge șirul inițial pentru a amplasa pe poziția curentă elementul corespunzător din șirul ordonat. Pentru a realiza amplasarea va trebui să căutăm cel mai mic element al subșirului rămas, deci vom realiza o parcurgere a acestui subșir. După identificarea elementului care trebuie amplasat în poziția curentă, vom realiza o interschimbare.

Varianta #1

Versiunea în pseudocod a algoritmului care realizează exact aceste operații este următoarea:

```
atr <— 0
comp <— 0
pentru i <— 1, n execută
 min <— a_i                          % atribuire %
  atr <— atr + 1
  ind <— i
  pentru j <— i + 1, n execută
    dacă min > a_j atunci             % comparare %
    min <— a_j                        % atribuire %
        atr <— atr + 1
        ind <— j
  sfârșit dacă
    comp <— comp + 1
    a_ind <— a_i                      % atribuire %
    atr <— atr + 1
    a_i <— min                        % atribuire %
    atr <— atr + 1
 sfârșit pentru
sfârșit pentru
```

Pentru realizarea interschimbării nu a fost nevoie de o variabilă auxiliară, variabila `min` având acest rol.

Se observă că, pentru un șir cu n elemente, numărul de comparări este același, indiferent de valoarea celor n numere.

Numărul atribuirilor poate varia doar datorită atribuirii din interiorul structurii alternative **dacă**.

Cazul cel mai favorabil apare atunci când această atribuire se execută de cele mai puține ori. Se observă că, în cazul în care șirul inițial este deja sortat, această atribuire nu este executată deloc, deci cel mai favorabil caz este cel în care șirul dat este deja sortat.

Cazul cel mai defavorabil apare atunci când atribuirea se execută de cele mai multe ori. Am putea crede că numărul maxim este $(n - 1) + (n - 2) + ... + 2 + 1 = n \cdot (n - 1) / 2$. Totuși, datorită permutărilor efectuate în cadrul algoritmului, nu putem găsi nici un șir pentru care această atribuire să fie executată de $n \cdot (n - 1) / 2$ ori.

La prima execuție a buclei **pentru** exterioare, bucla **pentru** interioară se execută de $n - 1$ ori, deci numărul maxim de execuții ale atribuirii este $n - 1$. Însă, după efectuarea permutării, pe ultima poziție ajunge cel mai mare element al șirului, deci la următoarea execuție a buclei **pentru** exterioare, numărul maxim de execuții ale atribuirii este $n - 3$. Este evident că, la fiecare pas, numărul de execuții se reduce cu 2.

Este ușor de găsit un șir care să ducă la o astfel de execuție a algoritmului: șirul trebuie să fie ordonat descrescător.

Se observă că nu există nici o altă configurație a șirului care să determine un număr mai mare de execuții ale atribuirii, deci putem considera că șirul ordonat descrescător reprezintă cazul cel mai defavorabil pentru acest algoritm.

În tabelul 1 este prezentată o statistică a numărului de atribuiri și comparări efectuate în diferite situații.

N	Favorabil		Mediu		Defavorabil	
	comp	atr	comp	atr	comp	atr
10	45	30	45	41	45	55
20	190	60	190	94	190	160
30	435	90	435	151	435	315
40	780	120	780	211	780	520
50	1225	150	1225	273	1225	775
60	1770	180	1770	336	1770	1080
70	2415	210	2415	401	2415	1435
80	3160	240	3160	467	3160	1840
90	4005	270	4005	533	4005	2295
100	4950	300	4950	600	4950	2800

Tabelul 1: SelectSort varianta #1

Varianta #2

Putem optimiza acest algoritm dacă, la fiecare pas, nu păstrăm valoarea celui mai mic număr, ci doar poziția acestuia. Se observă că numărul minim nu este suprascris decât după ce a fost determinat cu certitudine, minimele intermediare rămânând nemodificate.

Versiunea în pseudocod a algoritmului care realizează această mică optimizare este următoarea:

```
atr <— 0
comp <— 0
pentru i <— 1, n execută
   ind <— i
   pentru j <— i + 1, n execută
      dacă a_ind > a_j atunci          % comparare %
      ind <— j
   sfârșit dacă
      comp <— comp + 1
      aux <— a_ind                     % atribuire %
      atr <— atr + 1
      a_ind <— a_i                     % atribuire %
      atr <— atr + 1
      a_i <— aux                       % atribuire %
      atr <— atr + 1
   sfârșit pentru
sfârșit pentru
```

Pentru realizarea interschimbării în această versiune avem nevoie de o variabilă auxiliară, deoarece variabila `min` nu mai este folosită.

Și în acest caz, numărul comparărilor este constant. Se observă însă, că și numărul atribuirilor este constant deoarece în interiorul structurii alternative nu se mai execută atribuiri asupra elementelor vectorului. Așadar, pentru această variantă a algoritmului numărul comparărilor și atribuirilor nu variază în funcție de configurația datelor de intrare, deci nu există cazuri favorabile sau defavorabile.

În tabelul 2 este prezentată o statistică a numărului de atribuiri și comparări efectuate în diferite situații. Pentru a nu crea confuzii vom păstra toate cele șapte coloane ale tabelului anterior, chiar dacă informațiile sunt redundante.

N	Favorabil		Mediu		Defavorabil	
	comp	atr	comp	atr	comp	atr
10	45	30	45	30	45	30
20	190	60	190	60	190	60
30	435	90	435	90	435	90
40	780	120	780	120	780	120
50	1225	150	1225	150	1225	150
60	1770	180	1770	180	1770	180
70	2415	210	2415	210	2415	210
80	3160	240	3160	240	3160	240
90	4005	270	4005	270	4005	270
100	4950	300	4950	300	4950	300

Tabelul 2: SelectSort varianta #2

Analiza variantelor

Se observă că, pentru cele două variante numărul comparărilor este același, deci, din acest punct de vedere ele sunt la fel de performante.

Numărul atribuirilor efectuate de cea de-a doua variantă a algoritmului este egal cu cel efectuat de prima variantă în cazul cel mai favorabil. Așadar, din acest punct de vedere, al doilea algoritm este cel puțin la fel de performant ca și primul. În figura 1 este prezentat graficul numărului de atribuiri pentru cazul mediu.

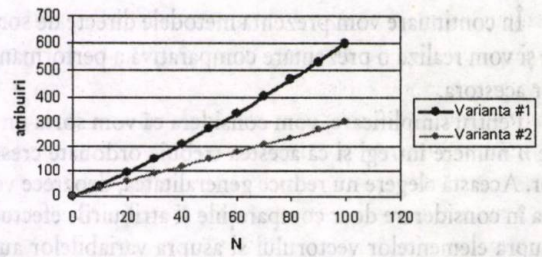


Figura 1: SelectSort: cazul mediu - atribuiri

Graficul numărului de atribuiri pentru cazul cel mai defavorabil este prezentat în figura 2.

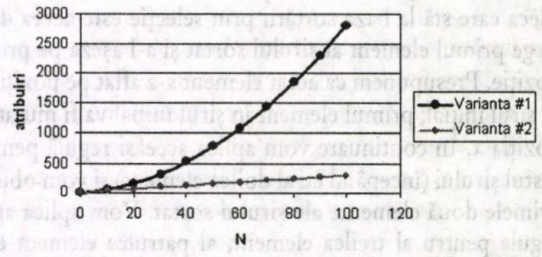


Figura 2: SelectSort: cazul cel mai defavorabil - atribuiri

Se observă clar că pentru varianta #1 graficul ia forma unei funcții pătratice, în timp ce pentru varianta #2 graficul are forma unei funcții liniare.

În figura 3 vom prezenta graficul numărului de comparări efectuate de algoritmul *SelectSort*. Numărul comparărilor nu variază în funcție de varianta folosită.

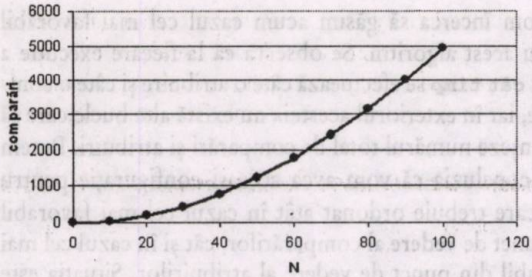


Figura 3: SelectSort: comparări

Din grafice putem trage concluzia că algoritmul *SelectSort* are un ordin de complexitate pătratic indiferent de structura datelor de intrare.

Sortarea prin inserare

Algoritmul de sortare prin inserare constă în parcurgerea șirului neordonat și inserarea elementului curent *i* în poziția corectă în subșirul de *i* - 1 elemente deja ordonat.

Mai exact, se începe cu elementul de pe poziția a doua și se verifică dacă este sau nu mai mic decât primul element. În caz afirmativ cele două elemente sunt interschimbate. După eventuala interschimbare suntem siguri că subșirul format din primele două elemente este ordonat. Se trece la al treilea element și se determină poziția în care trebuie inserat acesta în subșirul format din primele două elemente. Elementul este amplasat în poziția corectă, iar elementele care urmează după el sunt deplasate la dreapta cu o poziție. După inserare, subșirul format din primele trei elemente este sortat. Se continuă până la inserarea tuturor elementelor. În final vom obține un șir ordonat.

Există două variante importante ale algoritmului, fiecare având diferite subvariante. Diferența principală constă în modul în care este realizată căutarea: căutare ***liniară*** sau ***binară***.

Vom prezenta câte două subvariante pentru fiecare dintre modalitățile de căutare. Datorită faptului că algoritmul de căutare liniară este mai simplu, vom trata, pentru început, subvariantele în care poziția elementului curent este determinată folosind o căutare liniară.

Varianta #1 subvarianta #1

Vom prezenta acum o variantă în care se folosește un algoritm de căutare liniară pentru a determina poziția în care este inserat elementul curent aflat inițial pe o poziție *i*. Pentru aceasta vom parcurge șirul de la stânga la dreapta până vom găsi un element mai mare decât cel curent, iar apoi vom deplasa celelalte elemente (până la poziția *i* - 1) cu o poziție la dreapta și vom insera elementul curent în subșirul ordonat pe poziția corectă. Dacă nu există nici un element mai mare decât cel de pe poziția *i*, atunci poziția acestuia nu va mai fi modificată.

Pseudocodul algoritmului descris este următorul:

```
atr <— 0
comp <— 0
pentru i <— 2, n execută
   j <— 0
   cât timp a_j < a_i și j < i execută
         % comparare în buclă anterior condiționată%
      j <— j + 1
      comp <— comp + 1
   sfârșit cât timp
   comp <— comp + 1
   aux2 <— a_j                          % atribuire %
   atr <— atr + 1
   pentru k <— j, i - 1 execută
      aux1 <— a_k+1                     % atribuire %
      atr <— atr + 1
      a_k+1 <— aux2                     % atribuire %
      atr <— atr + 1
      aux2 <— aux1                      % atribuire %
      atr <— atr + 1
   sfârșit pentru
   a_j <— aux2                          % atribuire %
   atr <— atr + 1
sfârșit pentru
```

Trebuie remarcat faptul că pentru a deplasa elementele avem nevoie, la fiecare pas, de trei atribuiri.

Vom încerca să găsim acum cazul cel mai favorabil din punct de vedere al comparărilor efectuate. În cel mai favorabil caz corpul buclei **cât timp** nu ar trebui să se execute nici o dată. Pentru aceasta va trebui ca elementul curent să fie inserat întotdeauna pe prima poziție, deci șirul ar trebui să fie ordonat descrescător.

Cazul cel mai defavorabil apare atunci când corpul buclei **cât timp** se execută de cele mai multe ori, așadar elementul curent ar trebui să nu își modifice niciodată poziția, deci șirul trebuie să fie ordonat crescător.

Vom studia acum cazul cel mai favorabil și cel mai defavorabil pentru atriburi. În cel mai favorabil caz, corpul buclei interioare **pentru** nu trebuie să se execute nici o dată. Aceasta înseamnă că nu sunt necesare deplasări, deci elementele nu trebuie să își modifice poziția. Ca urmare, în cel mai favorabil caz șirul trebuie să fie ordonat crescător.

Cazul cel mai defavorabil apare în situația în care corpul buclei interioare **pentru** se execută de cele mai multe ori. Aceasta înseamnă că elementul curent ar trebui inserat întotdeauna pe prima poziție, deci șirul trebuie să fie ordonat descrescător.

Se observă că situația este oarecum bizară. Cazul cel mai favorabil pentru comparări este cazul cel mai defavorabil pentru atribuiri, iar cazul cel mai defavorabil pentru comparări este cazul cel mai favorabil pentru atribuiri.

Datorită faptului că nu poate fi determinată o echivalență absolută între un anumit număr de atribuiri și un anumit număr de comparări, nu există posibilitatea determinării cu exactitate a celui mai favorabil sau celui mai defavorabil caz. În tabelul 3 este prezentată o statistică a numărului de atribuiri și comparări efectuate în diferite

situații. Datorită situației prezentate, nu mai există coloanele corespunzătoare cazului celui mai favorabil și cazului celui mai puțin favorabil. Ele sunt înlocuite de coloane corespunzătoare unui șir ale cărui elemente sunt ordonate crescător și unui șir ale cărui elemente sunt ordonate descrescător.

N	Crescător		Mediu		Descrescător	
	comp	atr	comp	atr	comp	atr
10	54	18	31	86	9	153
20	209	38	113	326	19	608
30	464	58	244	716	29	1363
40	819	78	425	1259	39	2418
50	1274	98	655	1953	49	3773
60	1829	118	935	2799	59	5428
70	2484	138	1263	3798	69	7383
80	3239	158	1642	4947	79	9638
90	4094	178	2071	6244	89	12193
100	5049	198	2547	7702	90	15048

Tabelul 3: InsertSort varianta #1 subvarianta #1

Varianta #1 subvarianta #2

Se observă imediat o posibilitate de optimizare a algoritmului anterior. Putem păstra elementul a cărui poziție trebuie modificată într-o variabilă auxiliară și apoi vom putea deplasa elementele la dreapta cu o poziție, parcurgând subșirul deja ordonat de la dreapta la stânga. Astfel, pentru deplasări nu se vor mai efectua trei atribuiri la fiecare pas, ci doar una singură, deoarece interschimbarea este înlocuită de o singură atribuire.

De asemenea, putem determina poziția în care trebuie inserat elementul parcurgând subșirul ordonat de la dreapta la stânga. Așadar, vom parcurge elementele până vom găsi unul care este mai mic decât elementul curent sau ajungem pe prima poziție.

Pe măsură ce efectuăm căutarea, putem să efectuăm și deplasările.

În final, vom insera elementul curent în poziția determinată.

Pseudocodul acestui algoritm este prezentat în continuare:

```
atr <— 0
comp <— 0
pentru i <— 2, n execută
   j <— i
   aux <— a_i                    % atribuire %
   atr <— atr + 1
  cât timp aux < a_(j-1) și j > 1 execută
        % comparare în buclă anterior condiționată%
      comp <— comp + 1
      a_j <— a_(j-1)             % atribuire %
      atr <— atr + 1
      j <— j - 1
  sfârșit cât timp
   comp <— comp + 1
   a_j <— aux                    % atribuire %
   atr <— atr + 1
sfârșit pentru
```

Vom încerca să găsim acum cazul cel mai favorabil pentru acest algoritm. Se observă că la fiecare execuție a buclei **cât timp** se efectuează câte o atribuire și câte o comparare, iar în exteriorul acesteia nu există alte bucle care să influențeze numărul total de comparări și atribuiri. Putem trage concluzia că vom avea aceeași configurație pentru șirul care trebuie ordonat atât în cazul cel mai favorabil din punct de vedere al comparărilor, cât și în cazul cel mai favorabil din punct de vedere al atribuirilor. Situația este aceeași și pentru cazul cel mai defavorabil.

Cazul cel mai favorabil apare atunci când corpul buclei **cât timp** nu se execută nici o dată, deci poziția elementului curent nu se modifică. Așadar, șirul trebuie să fie ordonat crescător. Cazul cel mai defavorabil apare atunci când corpul buclei **cât timp** se execută de cele mai multe ori, deci elementul curent va fi inserat întotdeauna pe prima poziție. Așadar, șirul trebuie să fie ordonat descrescător.

În tabelul 4 este prezentată o statistică a numărului de atribuiri și comparări efectuate în diferite situații.

N	Favorabil		Mediu		Defavorabil	
	comp	atr	comp	atr	comp	atr
10	9	18	30	39	54	63
20	19	38	113	132	209	228
30	29	58	243	272	464	493
40	39	78	425	464	819	858
50	49	98	654	703	1274	1323
60	59	118	934	993	1829	1888
70	69	138	1264	1333	2484	2553
80	79	158	1642	1721	3239	3318
90	89	178	2070	2159	4094	4183
100	99	198	2550	2649	5049	5148

Tabelul 4: InsertSort varianta #1 subvarianta #2

Compararea subvariantelor

Se observă că, pentru cele două subvariante, numărul comparărilor este aproximativ același deci, din acest punct de vedere ele sunt la fel de performante. Singurele diferențe apar în cazul mediu, dar sunt nesemnificative și sunt datorate faptului că șirurile folosite pentru determinarea numărului de atribuiri și comparări în cazul mediu au fost generate aleator.

În cazul mediu și în cazul cel mai defavorabil numărul atribuirilor efectuate de cea de-a doua subvariantă a algoritmului este de aproximativ trei ori mai mic decât numărul atribuirilor efectuate de prima variantă. În cazul cel mai favorabil, numărul atribuirilor efectuate de cele două subvariante este identic. Așadar, din acest punct de vedere, al doilea algoritm este cel puțin la fel de performant ca și primul. În figura 4 este prezentat graficul numărului de atribuiri pentru cazul mediu.

Figura 4: InsertSort #1: cazul mediu - atribuiri

Graficul numărului de atribuiri pentru cazul cel mai defavorabil este prezentat în figura 5.

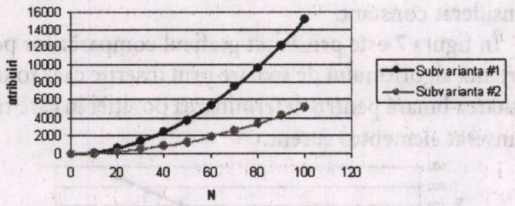


Figura 5: InsertSort #1: cazul cel mai defavorabil - atribuiri

Se observă că graficele numărului de atribuiri au forma unor funcții pătratice.

În figura 6 vom prezenta graficul numărului de comparări pentru cazul mediu și cazul cel mai defavorabil.

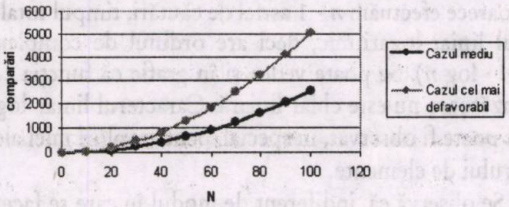


Figura 6: InsertSort #1: comparări

Din grafice putem trage concluzia că algoritmul *InsertSort* are un ordin de complexitate pătratic pentru cazul mediu și cazul cel mai defavorabil. În cel mai favorabil caz, cea de-a doua subvariantă a algoritmului rulează în timp liniar.

Varianta #2 subvarianta #1

Următoarea variantă pe care o prezentăm folosește un algoritm de căutare binară pentru a determina poziția în care este inserat elementul curent aflat inițial pe o poziție *i*. După determinarea poziției în care va fi inserat elementul curent, elementele cu valori mai mari sunt deplasate la stânga cu o poziție.

O versiune a pseudocodului acestui algoritm este:

```
atr <— 0
comp <— 0
pentru i <— 2, n execută
  stg <— 0
  dr <— i
  mijl <— [(stg + dr) / 2]
  cât timp stg < dr execută
    comp <— comp + 1
    dacă a_mijl < a_i                    % comparare %
      atunci stg <— mijl + 1
      altfel dr <— mijl - 1
  sfârșit dacă
  sfârșit cât timp
    dacă a_mijl < a_i                    % comparare %
      atunci mijl <— mijl + 1
  sfârșit dacă
  comp <— comp + 1
  aux2 <— a_mijl                         % atribuire %
  atr <— atr + 1
  pentru k <— mijl, i - 1 execută
    aux1 <— a_k+1                        % atribuire %
    atr <— atr + 1
    a_k+1 <— aux2                        % atribuire %
    atr <— atr + 1
    aux2 <— aux1                         % atribuire %
    atr <— atr + 1
  sfârșit pentru
  a_j <— aux2                            % atribuire %
  atr <— atr + 1
sfârșit pentru
```

Se observă că singura diferență față de prima subvariantă a primei variante este dată de modul în care se realizează căutarea. Aceasta se realizează în timp constant $O(\log i)$ unde i este indicele elementului a cărui poziție se caută și nu depinde de configurația șirului.

Există o mică variație nesemnificativă datorită faptului că în urma înjumătățirilor, diferența dintre numărul elementelor din cele două jumătăți ar putea fi 1, deci, în unele cazuri, s-ar putea ca numărul de pași necesari pentru găsirea pozițiilor să varieze. În urma studierii rezultatelor se va observa că, din nou, cazul cel mai favorabil din punct de vedere al comparărilor este identic cu cazul cel mai defavorabil din punct de vedere al atribuirilor și cazul cel mai defavorabil din punct de vedere al comparărilor este identic cu cazul cel mai favorabil din punct de vedere al atribuirilor. Totuși, diferențele sunt foarte mici și pot fi ignorate.

Deplasările se realizează în același mod, deci vom avea cazul cel mai defavorabil atunci când elementele sunt inserate pe prima poziție (șirul este ordonat descrescător) și cazul cel mai favorabil atunci când poziția elementului nu se modifică (șirul este ordonat crescător).

În tabelul 5 este prezentată o statistică a numărului de atribuiri și comparări efectuate pentru cazul cel mai favorabil, cazul mediu și cazul cel mai defavorabil.

N	Favorabil		Mediu		Defavorabil	
	comp	atr	comp	atr	comp	atr
10	28	18	26	86	24	153
20	73	38	68	326	64	608
30	123	58	117	716	112	1363
40	182	78	171	1259	162	2418
50	242	98	230	1953	216	3773
60	302	118	290	2799	276	5428
70	369	138	351	3798	336	7383
80	439	158	416	4947	396	9638
90	509	178	483	6244	456	12193
100	579	198	553	7702	522	15048

Tabelul 5: InsertSort varianta #2 subvarianta #1

Varianta #2 subvarianta #2

Vom efectua aceeași optimizare ca în cazul primei variante. Pentru efectuarea deplasărilor, vom parcurge subșirul ordonat de la dreapta spre stânga. Totuși, nu vom efectua simultan și comparările, deoarece căutarea nu mai este li-

niară, ci binară. Pseudocodul acestui algoritm este prezentat în continuare:

```
atr <— 0
comp <— 0
pentru i <— 2, n execută
   stg <— 0
   dr <— i
   mijl <— [(stg + dr) / 2]
  cât timp stg < dr execută
      comp <— comp + 1
      dacă a_mijl < a_i                    % comparare %
         atunci stg <— mijl + 1
         altfel dr <— mijl - 1
   sfârșit dacă
  sfârșit cât timp
      dacă a_mijl < a_i                    % comparare %
         atunci mijl <— mijl + 1
   sfârșit dacă
   comp <— comp + 1
   aux <— a_i                              % atribuire %
   atr <— atr + 1
   pentru k <— i - 1, mijl pas -1 execută
      a_k+1 <— a_k                         % atribuire %
      atr <— atr + 1
  sfârșit pentru
   a_mijl <— aux                           % atribuire %
   atr <— atr + 1
sfârșit pentru
```

Comparările sunt efectuate în același mod ca și în cazul subvariantei anterioare, deci numărul lor poate fi considerat constant.

Atribuirile sunt efectuate la fel ca în cazul celei de-a doua subvariante a primei variante, deci vom avea cazul cel mai favorabil atunci când șirul este ordonat crescător și cazul cel mai defavorabil atunci când șirul este ordonat descrescător.

Numărul atribuirilor și comparărilor pentru diferite situații apare în tabelul 6.

N	Favorabil		Mediu		Defavorabil	
	comp	atr	comp	atr	comp	atr
10	28	18	26	39	24	63
20	73	38	68	132	64	228
30	123	58	117	272	112	493
40	182	78	171	464	162	858
50	242	98	230	703	216	1323
60	302	118	290	993	276	1888
70	369	138	351	1333	336	2553
80	439	158	416	1721	396	3318
90	509	178	483	2159	456	4183
100	579	198	553	2649	522	5148

Tabelul 6: InsertSort varianta #2 subvarianta #2

Compararea subvariantelor

Din nou, pentru cele două subvariante numărul comparărilor este aproximativ același, deci, din acest punct de vedere ele sunt la fel de performante.

Numărul comparărilor nu variază, practic, în funcție de configurația șirului care trebuie ordonat, deci poate fi considerat constant.

În figura 7 este prezentat graficul comparărilor pentru varianta algoritmului de sortare prin inserție care folosește căutarea binară pentru determinarea poziției în care trebuie inserat elementul curent.

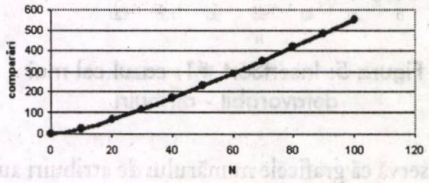


Figura 7: InsertSort #2: comparări

Fiecare căutare se realizează într-un timp logaritmic. Deoarece efectuăm n - 1 astfel de căutări, timpul total va fi unul liniar-logaritmic, deci are ordinul de complexitate $O(n \cdot \log n)$. Se poate vedea și în grafic că funcția corespunzătoare nu este chiar liniară. Caracterul liniar-logaritmic poate fi observat, în special, pentru valori mici ale numărului de elemente.

Se observă că, indiferent de modul în care se face căutarea, elementul curent va fi inserat pe aceeași poziție. Așadar, numărul atribuirilor pentru prima subvariantă este același indiferent de modul de căutare. Situația este aceeași și pentru cea de doua subvariantă. Așadar graficele care prezintă numărul atribuirilor sunt aceleași ca și în cazul primei variante.

Așadar, din punct de vedere al atribuirilor, putem trage aceleași concluzii ca și în cazul primei variante: algoritmul *InsertSort* are un ordin de complexitate pătratic pentru cazul mediu și cazul cel mai defavorabil.

Compararea variantelor

Vom compara acum cele două variante de realizare a algoritmului de sortare prin inserție.

Vom alege cea mai performantă dintre cele două subvariante a celor două variante și anume cea de-a doua.

Așa cum am arătat anterior, numărul atribuirilor nu diferă de la o variantă la alta, deoarece elementul curent va fi inserat în aceeași poziție și modul de inserare al elementelor este același.

Diferențe apar doar în cazul comparărilor datorită faptului că diferă modul de căutare al poziției în care este inserat elementul curent.

Din acest punct de vedere, prima variantă rulează în timp liniar pentru cazul cel mai favorabil și în timp pătratic pentru cazul mediu și cazul cel mai defavorabil. Cea de-a doua variantă rulează în timp liniar-logaritmic indiferent de configurația șirului.

Vom prezenta în continuare graficele numărului de comparări pentru cazul cel mai favorabil, cazul mediu și cazul cel mai defavorabil.

Pentru cazul cel mai favorabil, graficul comparărilor este cel din figura 8.

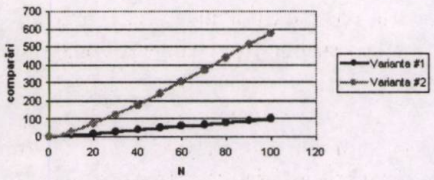


Figura 8: InsertSort: cazul cel mai favorabil - comparări

Este evident faptul că, în cazul cel mai favorabil, prima variantă este mai performantă. Aceasta se datorează faptului că o căutare liniară se realizează în timp $O(1)$ pentru cazul cel mai favorabil în timp ce căutarea binară necesită un timp $O(\log n)$.

Pentru cazul mediu, graficul comparărilor este prezentat în figura 9.

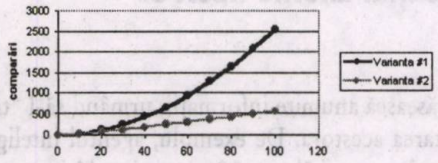


Figura 9: InsertSort: cazul mediu - comparări

În acest caz își face simțită prezența perfomanța superioară a algoritmului de căutare binară, față de algoritmul de căutare liniară. Chiar dacă pentru valori mici ale numărului de elemente ale șirului, numărul comparărilor nu diferă foarte mult de la o variantă la alta, pentru valori mari diferența este evidentă.

Pentru cazul mediu, căutarea liniară are un timp de execuție liniar (în medie, pentru determinarea poziției în care trebuie inserat al i-lea element al vectorului, se efectuează i / 2 comparări), în timp ce timpul de execuție al căutării binare este unul logaritmic. Aceasta este explicația motivului pentru care cea de-a doua variantă este mai performantă.

În figura 10 este prezentat graficul comparărilor pentru cel mai defavorabil caz.

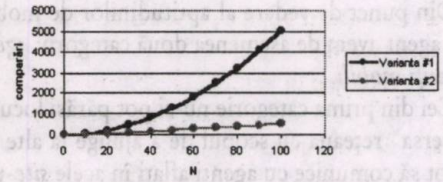


Figura 10: InsertSort: cazul cel mai defavorabil - comparări

Diferența dintre cele două metode de căutare este acum și mai evidentă. În acest caz, pentru determinarea poziției în care trebuie inserat al i-lea element al vectorului, se efectuează i - 1 comparări.

Așadar, cea de-a doua variantă a algoritmului este mai performantă în cazul cel mai defavorabil.

Putem trage concluzia că a doua variantă a algoritmului este, în majoritatea cazurilor, mai performantă decât prima variantă.

Concluzii

Probabil că v-ați dat seama că scopul acestui articol nu a fost prezentarea unor metode de sortare care, de altfel, sunt cunoscute de marea majoritate a cititorilor revistei noastre.

Am încercat să arătăm modul în care pot fi comparați diferiți algoritmi care rezolvă aceeași problemă pentru a-l alege pe cel mai performant.

De asemenea, am dorit să arătăm importanța detaliilor de implementare. În funcție de diferite alegeri pe care le facem putem îmbunătăți semnificativ performanțele.

Nu am prezentat demonstrații riguroase ale diferitelor afirmații pentru a nu încărca articolul cu elemente matematice mai greu de înțeles și, practic, neinteresante pentru majoritatea cititorilor.

Am prezentat câteva metode intuitive prin care se pot găsi ordinele de complexitate ale diferitelor variante ale unor algoritmi, fără a fi nevoie de un aparat matematic bine pus la punct.

În practică, sunt puțini cei care realizează demonstrații riguroase; majoritatea sunt mulțumiți de demonstrațiile intuitive și în foarte puține situații se dovedește că astfel de demonstrații sunt eronate.

Sfatul nostru este să încercați întotdeauna să folosiți cea mai performantă variantă a unui algoritm, iar dacă există mai mulți algoritmi cu ajutorul cărora poate fi rezolvată problema cu care vă confruntați, să îl alegeți întotdeauna pe cel care duce la rezultate corecte în timpul cel mai scurt.

Pentru aceasta nu este necesar să vă complicați folosind calcule matematice complexe, ci puteți desena doar câteva grafice și diferențele între algoritmi vor fi evidente în majoritatea cazurilor.

Chiar și simplele tabele pe care le-am folosit sunt suficiente pentru a vă da seama care dintre algoritmii sau variantele pe care le aveți la dispoziție va duce la obținerea celor mai bune performanțe.

Așa cum ați văzut, detalii minore de implementare pot îmbunătăți semnificativ performanțele. Așa cum am arătat în cazul sortării prin inserția, simpla modificare a direcției de parcurgere a șirului poate duce la reducerea de aproximativ trei ori a timpului de execuție.

Probabil că vă veți întâlni în multe situații cu necesitatea de a alege între doi sau mai mulți algoritmi. Vor exista algoritmi mai performanți în cazul mediu, dar mai puțin performanți în cazul cel mai defavorabil. Un exemplu în acest sens ar putea fi algoritmii *HeapSort* și *QuickSort*. Va trebui ca, întotdeauna, în funcție de situațiile particulare ale problemei, să puteți alege care dintre algoritmi este mai potrivit.

Mihai Scorțaru este redactor-șef adjunct al GInfo. Poate fi contactat prin e-mail la adresa **skortzy@yahoo.com**.

Agenți INTELIGENȚI

Ciprian Ileană

Ideea folosirii agenților nu este nouă. Ea a apărut pentru prima dată în lucrările lui Nicholas Negraponte (1970) și Alan Kai (1984). Alți cercetători cum ar fi Marvin Misky, Oliver Selfridge, Rodney Brooks, Pattie Mass au studiat problemele legate de agenți și au evidențiat diferite tipuri de comportament ale lor.

O posibilă definiție

Agenții inteligenți sunt programe adaptive și autonome, care sunt folosite în vederea creării de produse *software* care rezolvă anumite sarcini (*task*-uri) în numele unui anumit utilizator, bazându-se pe instrucțiuni explicite sau implicite din partea acestuia.

Agenții inteligenți pot îmbunătăți interacțiunea cu calculatorul prin:

- ascunderea complexității *task*-urilor dificile;
- realizarea de *task*-uri (acțiuni) laborioase;
- conducerea unor tranzacții în numele utilizatorului;
- pregătire și învățare;
- oferirea de ajutor unor anumiți utilizatori în vederea colaborării acestora;
- monitorizarea de evenimente și proceduri diverse.

Sarcinile pe care le poate realiza sau aplicațiile pe care le poate asista un agent sunt virtual nelimitate:

- filtrarea/selectarea informației;
- regăsirea informației;
- gestiunea corespondenței;
- planificarea întâlnirilor;
- selectarea cărților, spectacolelor, muzicii.

Tipuri de agenți inteligenți

Din punct de vedere al scopurilor pentru care au fost creați, deosebim două tipuri de agenți:

- Varietatea cea mai cunoscută de agent este cel construit cu scopul de a învăța și de a prelucra permanent sarcini de rutină sau repetitive ale utilizatorului. Acesta poate recunoaște, de exemplu, că utilizatorul ignoră în mod constant *e-mail*-urile care provin de la un anumit expeditor și le va șterge automat când acestea sosesc (utilizatorul nu află de primirea acestor scrisori elecronice, ele nefiind reținute în *Inbox*).
- O categorie mai specială și mai interesantă de agent este tipul *fire-and-forget*; un utilizator poate instrui un agent să găsească anumite informații, urmând să-l "trimită" în căutarea acestora. De exemplu, agentul inteligent astfel instruit poate fi "lansat" în rețea, iar el își poate croi drumul spre sistemul de computere al renumitei *Biblioteci a Congresului* (*Library of Congress in US*) cu scopul de a cerceta/întreba dacă și unde se poate afla informația de care utilizatorul are nevoie. Acesta este astfel eliberat de laborioasa sarcină de a căuta pe *Internet* informațiile care îl interesează, lucrul acesta fiind realizat, în schimb, de către agentul său, care după un timp (poate și câteva zile) îi va furniza rezultatul căutării. În acest timp, utilizatorul se poate concentra asupra unor probleme mai creative, îmbunătățindu-și astfel randamentul muncii. Sarcinile de rutină vor fi controlate și efectuate de către agent. Alte operațiuni pe care le poate efectua un agent inteligent tip *fire-and-forget* ar fi: programarea de întâlniri (*scheduling appointments*), realizarea de cumpărături pe piețele electronice (*e-commerce* sau *e-shopping* sau *virtual shopping*), sesizarea apariției anumitor oportunități (de care utilizatorul este interesat) și alertarea, eventual remedierea (automată) a problemelor de *software*.

Din punct de vedere al aptitudinilor de mobilitate ale unui agent avem de asemenea două categorii: *agenți statici* și *agenți mobili*.

Cei din prima categorie nu-și pot părăsi locul pentru a "traversa" rețeaua cu scopul de a ajunge la alte servere și nu pot să comunice cu agenți aflați în acele *site*-uri.

Agenții mobili, în schimb, pot efectua acest lucru, ei fiind caracterizați ca fiind obiecte ce posedă comportament, stare și locație. În vederea realizării sarcinii ce i s-a încredințat, un agent din această categorie poate părăsi *site*-ul pentru care a fost creat și poate naviga prin rețea, în fiecare moment el identificându-se printr-o anumită locație (*site*-ul pe care a ajuns), un anumit comportament (*task*-ul pe care îl realizează în acea situație/stare) și starea corespunzătoare.

Structura agenților inteligenți

Putem considera un agent inteligent ca fiind caracterizat de o arhitectură și un program.

Arhitectura realizează o interfață între percepțiile pe care agentul inteligent le obține din mediu și programul de care dispune agentul inteligent. Tot ea este cea care rulează programul, asigurând, în același timp, efectuarea diferitelor acțiuni alese, pe măsură ce acestea sunt generate. Arhitectura se poate realiza folosindu-se diferite tehnici din multitudinea celor existente, cum ar fi rețelele semantice, scenariile, cadrele etc.

Programul de care dispune agentul inteligent nu este altceva decât o funcție menită să realizeze corespondența dintre acțiunile intreprinse de către agentul inteligent și percepțiile pe care acesta le primește de la mediu. Foarte important este ca programul să fie compatibil din toate punctele de vedere cu arhitectura de care dispune agentul inteligent.

Dacă ținem cont de informația creată explicit și mai apoi folosită de către agent în cadrul procesului de decizie, avem de a face cu numeroase tipuri de agenți inteligenți. Programele agent se diferențiază între ele prin modalitatea de considerare a percepțiilor, acțiunilor, scopurilor și a mediului. De exmplu, avem mai jos câteva clase de agenți inteligenți:

- agenți reflex (agenți care se bazează pe reflexe simple);
- agenți care păstrează o imagine a lumii;
- agenți care se bazează pe scop;
- agenți care se bazează pe utilitate.

Agenți care se bazează pe reflexe simple

Un agent care se bazeză pe reflexe simple (agent reflex) se caracterizează prin răspunderea imediată la o secvență de intrare.

Un astfel de agent inteligent realizează conexiuni între situația curentă (situația pe care el o percepe) și o anumită acțiune. Aceste conexiuni sunt memorate de către agent sub forma unor reguli de producere sau reguli condiție-acțiune.

Programul de care dispune un agent reflex este unul foarte simplu. Acesta caută întotdeauna o regulă a cărei parte de condiție să se potrivească situației curente (așa cum este ea definită de către percepție), după care efectuează acțiunea pe care o are asociată cu această regulă.

Agenți care păstrează o imagine a lumii

Considerând condițiile exterioare precum și starea sa internă, un agent uman alege o anumită acțiune. În anumite tipuri de situații, unui agent artificial îi este necesară menținerea unei anumite informații cu privire la starea internă pentru a putea distinge între diferite stări ale lumii care generează secvențe perceptuale identice, dar care sunt, totuși, în mod esențial diferite (adică necesită acțiuni diferite).

Pentru a se putea modifica starea internă este necesar ca programul de care dispune agentul inteligent să includă două tipuri de informații (cunoștințe):

- informații despre modul în care lumea (universul, domeniul de competență al agentul inteligent) evoluează în mod independent față de agent;
- informații cu privire la modul în care acțiunile agentului inteligent afectează lumea.

Agentul inteligent este capabil să interpreteze o secvență perceptuală nouă (o situație nouă) prin prisma cunoașterii de care dispune în ceea ce privește lumea (codificată în starea internă). Informația cu privire la modul în care evoluează lumea este folosită de către agent pentru a păstra o urmă a părții nevăzute a lumii. Este necesar ca agentul să știe modul în care acțiunile sale interferează cu starea lumii.

Unui agent reflex i se pot asocia stări interne. În cazul acestei categorii de agenți, starea curentă se va combina cu starea internă, rezultând o nouă stare. Aceasta este considerată ca fiind noua stare curentă.

Acțiunea agentului inteligent se bazează pe căutarea unei reguli a cărei condiție satisface situația curentă, așa cum este ea definită de către secvența perceptuală și starea internă. În momentul găsirii unei reguli care satisface cerințele, se efectuează acțiunea care este asociată cu această regulă.

Agenți care se bazează pe scop

Agenții care se bazează pe scop acționează în sensul îndeplinirii scopului pe care îl au.

Nu întotdeauna este suficientă cunoașterea stării curente pe care o are mediul pentru a se putea selecta o acțiune corectă. În unele cazuri, agentul inteligent are nevoie și de o anumită informație cu privire la scop, informație care să descrie situațiile care sunt de dorit. Programul de care dispune agentul inteligent poate combina această informație cu informația despre rezultatele acțiunilor sale pentru a putea mai apoi alege acțiunile care conduc la scopul propus.

Agenți care se bazează pe utilitate

Nu întotdeauna este suficientă cunoașterea scopului pentru a se putea genera cea mai bună strategie (sau comportare) posibilă a unui agent inteligent. Scopul determină o distincție între stările care sunt considerate favorabile și cele care sunt considerate nefavorabile. Dacă asociem fiecărei stări o măsură a performanței, vom putea compara diferite stări, din punct de vedere al satisfiabilității sau utilității pe care acestea o prezintă din punctul de vedere al agentului inteligent.

Utilitatea nu este altceva decât o funcție U care este definită pe o mulțime S a stărilor cu valori în mulțimea numerelor reale: $U: S \rightarrow R$

Funcția de utilitate permite agentului inteligent să ia decizii raționale în cazul scopurilor conflictuale. În cazul în care pot exista mai multe scopuri, dintre care nici unul nu poate fi atins cu certitudine, funcția de utilitate este cea care furnizează o metodă prin care probabilitatea de succes și importanța scopurilor să poată fi comparate.

Ținând cont de cele spuse mai sus, putem admite că utilitatea unei stări măsoară gradul de satisfacție pe care o are agentul inteligent, dacă atinge starea respectivă. Prin urmare, un agent inteligent, care se bazează pe utilitate, tinde să-și maximizeze propria sa satisfacție (fericire).

Mediul de lucru al agenților inteligenți

Un agent inteligent efectuează acțiuni în cadrul unui mediu dat, iar acesta îi furnizează informații (prin intermediul percepțiilor).

Există mai multe tipuri distincte de medii. Fiecare dintre acestea necesită un anumit tip de program al agenților inteligenți care acționează în el. Mediile pot fi descrise printr-o serie de caracteristici, după cum se poate observa în ceea ce urmează.

Medii accesibile/medii inaccesibile

Un mediu este accesibil unui agent inteligent, dacă aparatul senzorial al respectivului agent îi permite acestuia accesul complet la stările mediului. Un mediu este efectiv accesibil în cazul în care senzorii agentului inteligent detectează toate aspectele relevante în alegerea unei acțiuni. Un agent care acționează într-un mediu accesibil nu are nevoie de o stare internă care să conserve informații despre stările anterioare.

Medii deterministe/medii nedeterministe

Un mediu este determinist (din punctul de vedere al unui agent inteligent), dacă starea următoare a mediului este determinată complet de către starea curentă în care se găsește mediul și de acțiunile selectate de către respectivul agent inteligent. În cazul situațiilor reale, cele mai multe sunt atât de complexe încât, deși ele sunt complet deterministe, pentru scopuri practice trebuie însă tratate ca fiind nedeterministe

Mediu static/mediu dinamic

Un mediu este dinamic pentru agentul inteligent în cazul în care acesta se schimbă în timp ce agentul deliberează. În caz contrar spunem că respectivul mediu este static. În cadrul unui mediu dinamic, agentul inteligent trebuie să țină seama de schimbările intervenite în timp ce el alege o acțiune.

Mediu discret/mediu continuu

Un mediu este discret în cazul în care acesta este caracterizat de un număr finit de stări, acțiuni și percepții. În caz contrar spunem că mediul este continuu.

Mediu episodic/mediu neepisodic

În cadrul unui mediu episodic, experiența agentului este divizată în episoade. Un episod nu depinde de acțiunile ce au apărut în vreunul din episoadele anterioare.

Mediile care sunt inaccesibile, dinamice, continue și neepisodice reprezintă cazurile cele mai dificile pentru un agent inteligent. Un *controler* pentru un sistem complex sau un instructor interactiv pentru învățarea unei limbi străine sunt agenți inteligenți care acționează în cadrul unui astfel de mediu.

Modul de lucru al agenților inteligenți

Agenții inteligenți operează în cadrul unui mediu *software* cum ar fi sistemele de operare, bazele de date sau rețelele de calculatoare.

Tehnologia aflată în spatele agenților inteligenți este o combinație de tehnici din domeniul inteligenței artificiale și de metodologii de dezvoltare a sistemelor, cum ar fi programarea orientată pe obiecte, care permite programelor atât să învețe cât și să reacționeze la mediul înconjurător. Agenții inteligenți interacționează cu mediul din care fac parte prin criterii de selecție a datelor bazate pe reguli. Un agent inteligent își dezvoltă regulile corespunzătoare prin intermediul instrucțiunilor explicite furnizate de utilizator, prin imitarea utilizatorului, prin feedback (răspuns) pozitiv sau negativ primit de la utilizator și prin indicații obținute în urma interacționării cu agenți inteligenți.

Modul de comunicare al agenților inteligenți

Tipul interfeței sistemului pe care o are un agent inteligent depinde de tipul resurselor sistemului cu care el comunică. Tipurile de agenți inteligenți disponibili în prezent pot comunica doar cu resursele sistemului din mediul *software* aflat în imediata sa vecinătate. Astfel, agenții *MicroSoft Intellisense* pot opera doar în cadrul programelor *Microsoft*; agenții *Maxims* ai firmei *Apple* pot opera doar în cadrul sistemului de operare *System 7*. Cu toate acestea, *General Magic* dorește să schimbe acest lucru, astfel ca limitarea impusă de o anumită platformă să dispară. Se dorește stabilirea interfeței sistem *Telescript* ca standard pentru toate platformele de calculatoare și pentru toate protocoalele de comunicare. *Telescript* este un limbaj de programare, dezvoltat de *General Magic*, care permite agenților inteligenți să se miște liber în cadrul unei rețele, în vederea realizării unei anumite sarcini, și dă agenților inteligenți posibilitatea de a coinduce tranzacții și de a învăța de la orice sistem de calculatoare sau de la orice alt agent pe care-l întâlnește. Adoptarea unanimă a standardului *Telescript* va permite crearea unui univers virtual global. Cu cât este mai larg universul, cu atât va crește abilitatea agenților inteligenți. *General Magic* este un consorțiu al *AT&T*, *Apple Computer*, *Motorola*, *Matushita Electric Industrial*, *Sony* și *Philips Electronics*.

Sisteme multi-agent

Cercetările cu privire la rezolvarea problemelor utilizând agenți inteligenți au relevat existența multor probleme complexe, cum ar fi cele de decizie, planificare etc., care nu pot fi rezolvate de către agenți inteligenți izolați. În concluzie, pentru rezolvarea unor astfel de probleme s-a propus utilizarea ***sistemelor multi-agent*** (*SMA*).

Paradigma *sistemelor multi-agent* este bazată pe comportarea socială inteligentă, care se realizează prin interacțiuni între membrii inteligenți ai societății, aceștia fiind numiți *agenți cognitivi*. Agenții inteligenți interacționează pentru a realiza scopuri sociale care depășesc cu mult posibilitățile de care dispune un singur agent inteligent. Interacțiunile care au loc între agenți sunt posibile datorită comunicării între ei.

În cadul oricărui *SMA*, comunicarea este cea care joacă un rol esențial. Schimbul de mesaje pe care îl realizează agenții este realizat în deplină conformitate cu un anumit protocol prestabilit. Protocoalele de comunicare între agenți inteligenți sunt complicate, incluzând în același timp atât operații de nivel elmentar cât și ecuații de ordin mai înalt. Trimiterea de mesaje, primirea sau confirmarea mesajelor sunt operații elementare. Operațiile de nivel înalt includ acțiuni de tipul influențare, convingere, detectarea relațiilor inter-agent.

Prin intermediul protocoalelor sofisticate de comunicare între agenții inteligenți, diferiți agenți heterogeni (reprezentând surse diferite de cunoaștere) interacționează cu scopul rezolvării unor probleme complexe care necesită experiența lor combinată.

Principalele arii de utilizare a agenților inteligenți

Cercetarea guvernamentală, administrarea rețelelor, sunt doar câteva exemple ale multiplelor și variatelor arii de utilizare a agenților inteligenți, *software* pentru consumatori (recent), cât despre rentabilitatea lor: *Ovum*, o societate londoneză de consultanță, aprecia creșterea pieței agenților inteligenți de la 800 de milioane de dolari în anul 2000 la peste 1.2 miliarde de dolari în anul 2001.

Guvernul și instituțiile sponsorizate de *Guvern* (este vorba despre *Guvernul Statelor Unite ale Americii*) se numără printre cei mai activi sponsori ai domeniului agenților inteligenți, mai ales în domenii de mesaje, automatizarea procesului de muncă și regăsirea informației. De exemplu, agenții inteligenți sunt utilizați în mecanismele de urmărire și control a sofisticatelor mecanisme de luptă care presupun lansări de bombe (proiectile) (*missile systems*) cum ar fi platformele *Patriot* și *Tomahawk*. De asemenea, un volum mare de muncă a fost necesar pentru ca platforma informațională să permită lucrul cu agenți inteligenți.

Administrarea rețelelor (*network management*) este una dintre ariile în care agenții inteligenți sunt cei mai dezvoltați. A fost nevoie de aceștia pentru a monitoriza și măsura în mod continuu activitatea rețelei, presupunând și raportarea problemelor software ce intervin pe parcursul desfășurării procesului. Printre cele mai avansate categorii de agenți inteligenți destinați administrării rețelelor se numără produsele *In-Touch* ale firmei *IBM* și *Cloud* ale firmei *AT&T*. Acești agenți, nu numai că pun complet, oriunde și oricând comunicația sub controlul cumpărătorului, dar realizează și *task*-uri de genul plăți, servicii client, înregistrări și servicii director. Alte companii care oferă agenți pentru rețele sunt: *Synoptics*, *Alcatel*, *British Telecom* și *Nippon Telegraph&Telephone*.

Un exemplu de agent inteligent destinat aplicațiilor consumator este *PersonaLink* al firmei *AT&T*. El furnizează o platformă din care utilizatorii își pot lansa proprii agenți inteligenți într-un univers virtual administrat de *AT&T*. Acest univers include piețe în care agenții pot face tranzacții, magazine în care agenții pot selecta produse și servicii precum și biblioteci de unde agenții pot localiza și extrage informații. *PersonaLink* este prima aplicație care a utilizat standardul *Telescript*. Acest agent poate fi accesat printr-un agent digital personal, un dispozitiv de comunicare (ca un receptor) al firmei *Sony*. Recent s-au dezvoltat și interfețe prietenoase pentru introducerea comenzilor către agenți.

Câteva exemple de agenți inteligenți

Norns go to war este un proiect de colaborare între firma *CyberLife* și *DERA* (*Defense Evaluation and Research Agency* a *Guvernului Britanic*) destinat creării de agenți care să se comporte ca adversari în misiunile de pilotaj în simulatoarele de zbor. Departamentul militar dorește crearea unor piloți (virtuali) de elită, adică agenți/piloți *software* care să mențină în aer avioanele indiferent de atac și să poată urmării țintele perioade lungi de timp. De asemenea, *Ministerul Apărării al UK* (*United Kingdom - Regatul Unit al Marii Britanii*) împreună cu *CyberLife* au inițiat un contract pentru construirea unui aparat de zbor militar simulat controlat în întregime de un agent *software*. El va fi capabil, printre altele, să ia decizii raționale în funcție de context pentru a-și finaliza misiunea cu succes.

DICA (*Do-I-Care Agent?*), dezvoltat la *University of California at Irvine* tratează problema regăsirii resurselor pe *Web*. O dată ce s-a găsit un *site* interesant pe *Web*, cum se poate ști când a fost adăugat material nou și interesant pe acel *site*? Soluția *DICA* constă în verificarea/vizitarea periodică a *site*-ului și informarea utilizatorului numai în cazul când a fost adăugat ceva nou și interesant. Pentru aceasta, utilizatorul trebuie să furnizeze răspunsuri (*feedback*) lui *DICA* pentru "a-l învăța" ce înseamnă "interesant" din punctul de vedere al utilizatorului.

ReferralWeb creează automat rețele sociale reprezentând relații între prieteni, colegi și parteneri de afaceri, din informațiile publice disponibile pe *Web*. Două persoane sunt relaționate (există un *link* între ele) dacă sistemul a obținut informații că cele două persoane sunt dintr-un anumit punct de vedere relaționabile. Rețeaua poate fi astfel folosită pentru a ghida căutarea de informații, de experți. De exemplu, cineva dorește să utilizeze sistemul pentru a găsi specialiști în criptografie care este posibil, să se afle în apropiere, fără a-i fi cunoscuți. Sistemul va găsi experții și îi va prezenta solicitantului precizând în ce fel sunt relaționați, permițându-i acestuia să găsească informații de încredere de la experți de încredere, care sunt dispuși să ajute, deoarece sunt prieteni ai prietenilor acelei persoane.

Dintre numeroasele exemple întâlnite în literatura de specialitate mai amintim doi agenți care realizează sarcini de asistent personal în gestionarea poștei electronice și selectarea informațiilor electronice.

Maes (1994) a realizat un agent numit *Maximus*, care gestionează poșta electronică, învață să stabilească prioritățile, să șteargă și să retransmită mesaje, să sorteze și să arhiveze mesaje. În acest caz, agentul "privește peste umăr" atunci când utilizatorul utilizează poșta electronică. Agentul memorează perechile (situație, acțiune) generate când utilizatorul lucrează singur.

Agentul *NewT* realizează filtrarea știrilor sosite prin poșta electronică. Acest agent inteligent este implementat în C++ pe o platformă *Unix* și este un sistem care facilitează filtrarea știrilor furnizate de către *Unsenet Netnews*.

Concluzie

Câteva considerente în legătură cu agenții inteligenți:
- agenții inteligenți modifică stilul interacțiunii om-calculator;
- natura lor autonomă și automatică a fost deja acceptată;
- securitate (agenții inteligenți sunt asemănători virușilor);
- probleme cu privire la responsabilitatea legală;
- vor înlocui agenții inteligenți oamenii?

Ca parte finală, putem afirma că, agenții inteligenți reprezintă o soluție inteligentă pentru necesitățile mereu crescânde ale unui număr tot mai mare de utilizatori, funcțională, flexibilă (relativ independentă și adaptabilă diferitelor platforme și sisteme de comunicare). Prin automatizarea sarcinilor de lucru supărătoare prin faptul că presupun un grad înalt de consum de timp sau presupun acțiuni repetitive din partea utilizatorului unui calculator, agenții inteligenți eliberează utilizatorii de treburile acestea, permițându-le în schimb realizarea de *task*-uri mai valoroase pentru aceștia. Agenții vor fi cei ce realizează *task*-urile obositoare și plictisitoare și o vor face și mult mai repede, mai ieftin și mai eficient (cu erori mai puține) decât partea umană.

Gama atribuțiilor ce pot fi încredințate agenților inteligenți este practic nelimitată și include filtrarea poștei electronice (filtre *software* pentru serviciul de *e-mail*), căutarea în baze de date și realizarea de cumpărături în magazine virtuale (electronic *marketplaces*).

Bibliografie

[1]. Dumitrescu D., *Algoritmi genetici + Strategii Evolutive + Programare Evolutivă = Calcul Evolutiv*, Casa de Editură Albastră, Cluj Napoca, 1999.

[2] Maes P., *Agents' that reduces work and information overload*, Comm. ACM, 1994.

[3] Ioan Ileană, *Curs de Clasificarea și recunoașterea formelor*, Ministerul Educației și Cercetării Universitatea "1 Decembrie 1918", Alba Iulia, 2001

[4] Simpson Patrick K., *Artificial Neural Systems, Foundations, Paradigms, Applications and Implementations*, Pergamon Press, 1990.

[5] Dumitrescu D., *Principiile Inteligenței Artificiale*, Casa de Editura Albastră, Cluj Napoca, 1999.

[6] University of Calgary, *Intelligense Sistems Group*, http://www.ucalgary.ca

Ciprian Ileană este student în anul doi la Facultatea de Matemetică și Informatică din cadrul Universității Babeș Bolyai din Cluj-Napoca. Poate fi contactat prin e-mail la adresa ciprian_il@yahoo.com.

Ofertă specială

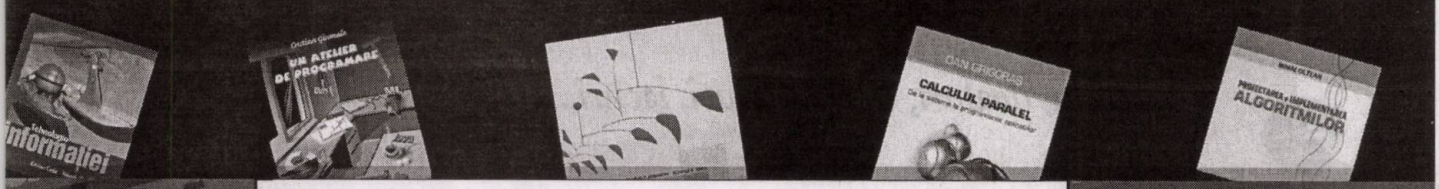
Începând cu sfârșitul lunii ianuarie, veți putea intra în posesia celebrei cărți editate de MIT Press, *Introduction to Algorithms*, scrisă de Thomas H. Cormen, Charles E. Leiserson și Ronald R. Rivest, a cărei traducere în limba română a apărut în anul 2000 la editura Computer Libris Agora din Cluj-Napoca.

Până la data de 28 februarie 2002 veți putea achiziționa această carte, direct de la editură, la un preț promoțional de 495.000 lei. O puteți comanda prin e-mail, telefonic sau în scris. Cartea va fi disponibilă și în librării, dar fără reducerea oferită de Computer Libris Agora.

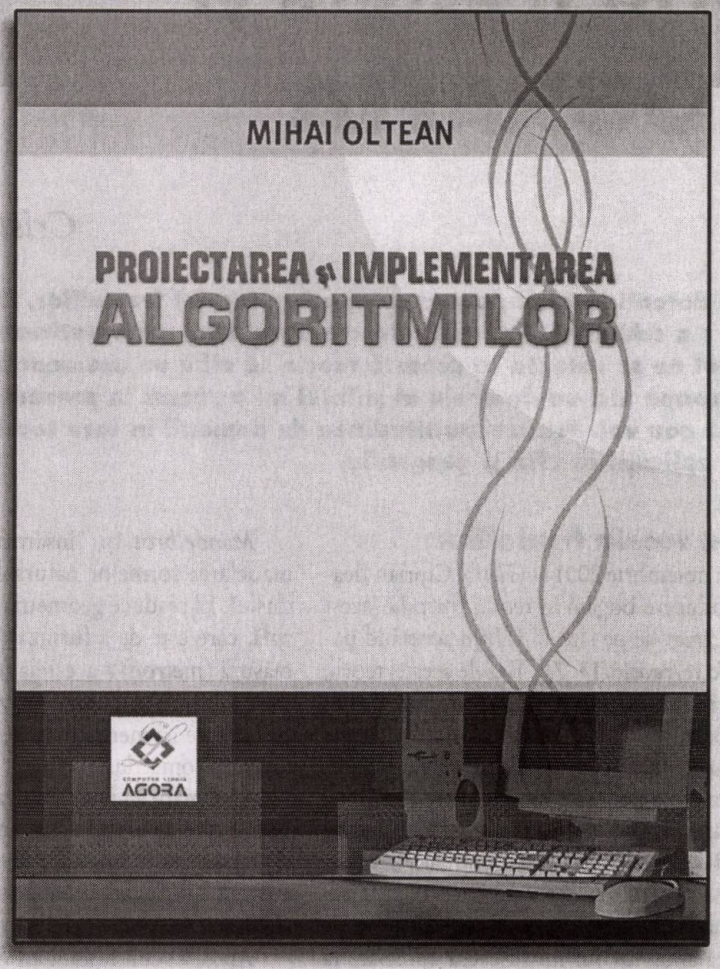
* * *

Această lucrare este recomandată tuturor celor care doresc să își îmbogățească cunoștințele de algoritmică, în special celor care se pregătesc pentru a participa la olimpiadele de informatică.

Cartea conține 882 de pagini, este structurată în 7 părți și 37 de capitole. Peste 260 de figuri și foarte mulți algoritmi prezentați în pseudocod formează un ansamblu bogat din care vă veți putea inspira oricând veți căuta să rezolvați probleme de algoritmică.



COMPUTER LIBRIS AGORA



"Cum aș putea să justific existența acestei cărți? Cum aș putea spre exemplu să justfic cele 100 de pagini despre programarea dinamică, acum când această metodă nu mai figurează în programele școlare? Nu știu. Sau, cum aș putea să justific capitolul despre algoritmi genetici, când pe plan mondial abia de câțiva ani a fost dovedită utilitatea și eficiența lor? Din nou nu știu! Dar rămâne bănuiala: a scrie algoritmi înseamnă a crea. Și nu poți crea fără să gândești."

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

Teoria fractală și GEOGRAFIA

Cristian Kalanyos

Cu câteva decenii în urmă, odată cu apariția teoriei fractalilor, întreaga lume științifică s-a schimbat. Probabil nici creatorul său, matematicianul Benoit Mandelbrot nu se aștepta ca această teorie să aibă un asemenea succes. Astfel, aproape nici un domeniu al științei nu a rămas în prezent neatins de către acest nou val. Printre multitudinea de domenii în care teoria fractalilor și-a găsit aplicații se află și geografia.

Introducere în teoria fractalilor

În numărul din luna noiembrie 2001 al *GInfo*, Ciprian Ileană a realizat o introducere bogată în *teoria fractală* (acest articol poate fi descărcat de pe site-ul *GInfo* accesând pagina **http://www.ginfo.ro/revista/11_7/**). Bazele acestei teorii, apărută relativ nou în lumea științifică, au fost puse de către *Benoit Mandelbrot*.

După apariția articolului său *Obiecte fractale* (1975), *Mandelbrot* își va expune pe larg teoria în anul 1982, publicând cartea sa manifest *The Fractal Geometry of Nature* (*Geometria fractală a naturii*). Această lucrare i-a conferit o audiență suficientă pentru a putea determina comunitatea științifică internațională să recunoască faptul că originalitatea și utilitatea spuselor lui nu fuseseră apreciate la adevărata lor valoare până în acel moment. Teoria fractală nu venea să completeze una deja existentă, ci prezenta un subiect de sine stătător și, ceea ce este cel mai important, odată cu ea apărea o nouă disciplină științifică cu totul aparte.

După cum spunea chiar autorul ei "*teoria fractală adaugă numeroase litere noi în alfabetul pe care Galilei l-a moștenit de la Euclid*".

În viziunea lui *Mandelbrot*, cea mai mare parte a formelor naturale nu pot fi reprezentate adecvat utilizându-se doar noțiunile pe care ni le oferă geometria euclidiană. Numeroase forme din natură sunt mult prea fragmentate sau neregulate, astfel încât complexitatea naturii o depășește cu mult pe cea a geometriei euclidiene, deoarece este limpede pentru oricine că norii nu sunt sfere, munții nu sunt conuri, țărmurile nu sunt cercuri, scoarța nu este netedă, iar fulgerul nu se propagă în linie dreaptă.

Mandelbrot își "însărcinează" geometria fractală cu modelarea formelor naturale utilizate de către geometria clasică. El readuce geometria în matca și la vocația ei originală, care este de a furniza o descriere a *Terrei* (*geo*) și o măsură (*metron*) a acesteia (dacă ne încredem în etimologie). Din multe puncte de vedere geometria fractală este prima care își merită numele, deci care urmează însuși demersul geometric.

Matematicienii *B. Riemann* și *N. Lobacevski* au fost primii care au lovit în fundamentele geometriei *euclidiene*, ei punând la îndoială cel de-al cincilea postulat din *Elementele* lui *Euclid*, cel al paralelelor, păstrând însă categoriile fundamentale, cum ar fi linia, suprafața etc.

Apoi, a venit rândul lui *Mandelbrot* să pornească, prin intermediul geometriei fractale, o adevărată revoluție devastatoare împotriva geometriei euclidiene. Geometria fractală respinge chiar și aceste categorii pe care se bazează postulatele *Elementelor* lui *Euclid*, contestând însăși substanța noțiunilor utilizate în axiomele și teoremele geometriei euclidiene.

Aplicarea fractalilor în geografie

Odată cu apariția lor, fractalii și-au găsit largi domenii de aplicare. Ei sunt acum întâlniți în *matematică*, *medicină* sau chiar *cinematografie*. Unul dintre primele domenii care a adoptat teoria fractală este *geografia*.

Această hotărâre s-ar părea să se justifice prin marele ajutor adus de aceștia în meteorologie, seismologie, geomorfologie și alte domenii adiacente geografiei.

Posibilitatea asocierii unor forțe similare care au puncte de aplicare la niveluri diferite ale scării, asocierea unui

set de date a unei curbe fractale, construirea de sisteme intrinseci, imprevizibile și haotice, înglobarea iregularităților ca părți esențiale ale modelului în locul tratării acestora ca zgomote care trebuie eliminate sunt doar câteva motive pentru care fractalii și-au găsit locul și în cadrul geografiei.

Deoarece tehnicile liniare nu oferă nici o modalitate de caracterizare, în cazul modelelor cu cicluri îmbinate unele în altele, s-a trecut la utilizarea de curbe fractale aproximate, cu ajutorul cărora se poate modela mai ușor acest tip de modele. Deși prezintă mari dezavantaje la nivel punctiform, utilizarea fractalilor ne sugerează mult mai exact caracterul esențial al curbei. De observat este faptul că, modelele meteorologice bazate pe fractali nu sunt modele anticipative.

Cutremurele au forme autoreflective deoarece, atât undele de avertizare care le preced, cât și replicile care le urmează, nu sunt altceva decât niște cutremure în miniatură. Deoarece cutremurul principal este reprezentat de o perioadă de intensă activitate, incluzând aici și perioade intense similare cu mici sincope între ele, el face parte din grupul fenomenelor greu de modelat cu ajutorul curbelor tradiționale. Astfel s-a ajuns la adoptarea fractalilor și de către seismologie.

Caracteristicile formelor geografice

Conceptul de formă este, în general, greu de exprimat, lucru observat mai ales în evaluarea formelor cu ajutorul geografiei. Una dintre cele mai utilizate măsuri în cadrul evaluării formelor geografice este reprezentată de către unitatea spațială.

Până acum nu a fost găsită nici o măsură de compactare care să fi rămas definitivă și, în același timp, să fie adecvată evaluării tuturor caracteristicilor geometrice. Unelte adiționale, care pot fi folosite pentru sporirea preciziei în natura spațială a compactării, sunt asigurate de aplicarea *teoriei fractale*.

Formele geografice constituie un concept dificil, atât din punct de vedere al exprimării, cât și al măsurătorilor, deoarece, în general, ele nu sunt drepte, circulare sau dreptunghiulare, ci au o natură mult mai complexă pentru a cărei descriere nu este suficientă utilizarea unei funcții analitice simple. Descrierea lor este o sarcină dificilă; în acest sens literatura de specialitate conține numeroase tehnici și metode de evaluare a formei (*Longley* - 1989, *Niemi* - 1990, *Goodchild* - 1987 ș.a.).

În cadrul descrierii formelor este des întâlnită caracteristica dată de către ***gradul de compactare***; în practică, această caracteristică este foarte rar evaluată printr-o măsurătoare specifică.

Una dintre tehnicile care pot fi aplicate în evaluarea caracteristicilor spațiale este cea fractală. Cea mai importantă investigație de până acum este cea realizată de către ***Mandelbrot*** în vederea măsurării ***adevăratei*** lungimi a liniei de coastă a *Marii Britanii*; el a aplicat analiza fractală asupra formelor geografice și a obținut gradul de compactare a formei.

Gradul de compactare

În ceea ce privește măsurarea gradului de compactare, dintre toate investigațiile care au avut loc până în prezent, trei merită luate în seamă, și anume *Niemi* - 1990, *Flaherty* - 1992 și *Horn* - 1993. Aceștia au identificat peste 20 de etaloane de compactare. Cea mai des implementată tehnică de evaluare a gradului de compactare este una bazată pe dezvoltarea unui raport de distanțe geometrice sau atribute de zone, distincte; acest raport a fost ulterior utilizat pentru a reprezenta indexul unei măsurări a gradului de compactare. Două dintre măsurătorile convenționale ale gradului de compactare au fost utilizate pentru evaluare. Măsurile următoare au fost selectate pentru a focaliza evaluarea asupra unor măsuri distincte, dar des întâlnite.

- Măsura de compactare *C*:
  - ♦ aria cercului înscris;
  - ♦ aria cercului circumscris;
- Măsura de compactare *F*:
  - ♦ circumferința cercului de arie egală;
  - ♦ perimetrul unei arii.

Prezența în cadrul peisajului natural a centrelor de populație cu dimensiuni variate constituie o caracteristică inevitabilă a organizării spațiale a activității umane. Influențe care pot schimba caracterul geografic al formelor geografice pot fi și factorii de origine istorică sau fizică. Linia de demarcație a unei granițe poate avea caracteristici și atribute care influențează forma și, în același timp, intră în calcul în momentul evaluării unei forme. De exemplu, o formă geografică poligonală poate consta din granița unui stat, însă, la fel de bine, poate fi și linia de demarcare a unui oraș. În componența granițelor formelor geografice intră deseori și demarcațiile naturale, astfel râurile și liniile de coastă asigură adesea o graniță naturală cu aspect sinusoidal, lucru ce contribuie la personalizarea formei geografice.

Pe parcursul procesului de evaluare a gradului de compactare a formelor geografice, este posibil ca cel ce evaluează să fie nevoit să se concentreze, mai degrabă, asupra anumitor porțiuni ale formei poligonului decât asupra întregului. De exemplu, în cazul studierii de granițe politice este posbil să fie necesară înlăturarea râurilor și a liniilor de coastă din cadrul evaluării. Această abilitate de înlăturare a porțiunilor din cadrul unei granițe sau de focalizare a evaluării asupra unei anumite zone, conferă un plus de flexibilitate în încercarea de a evalua o formă geografică în deplinătatea sa. Din păcate, apare însă o problemă, și anume înlăturarea granițelor nu poate fi pe deplin implementată utilizând măsuri convenționale de compactare din moment ce trebuie considerată întreaga zonă. Prin înlăturarea de granițe semnificative se poate ajunge la imposibilitatea calculării pentru că parametrii necesari nu pot fi colectați. În 1993 *Horn* a propus aplicarea modificării direct la nivelul datelor cu privire la graniță pentru a compensa în cazul geografiei insulelor și a peninsulelor; această modificare manipulează datele mai mult decât o făcea foca-

lizarea analizei și, drept urmare, rezultatele, în cazul tehnicilor de manipulare a datelor, pot induce erori imposibil de prevăzut.

Măsurători fractale

Cel mai adesea, formele geografice sunt complexe. Ele au fost descrise mediatic, public și juridic ca *serpentine*, *bacterii în creștere*, *păianjeni*, *forme cu colțuri răsucite și cranii* sau ca *fragmente ADN* (**Smothers** - 1993). Limbajul expresiv folosit în descrieri (utilizate pentru a relata caracteristicile formei) caracterizează formele geografice ca fiind de natură complexă, iar caracteristicile formelor descrise de către acest limbaj sunt, în mod normal, studiate prin intermediul analizei fractale (**Mandelbrot** - 1967). Cel care a investigat formele geografice complexe și geometria lor fractală a fost **Mandelbrot** - 1982, el dezvoltând moduri de abordare și algoritmi care analizează forme neregulate. La asigurarea existenței unei modalități de măsurare a proceselor regulare, ce apar pe o mare întindere de scări, contribuie noțiunea de ***dimensiune fractală***.

Aceasta a fost descoperită de către matematicienii ***Felix Haussdorff*** și ***Abram S. Besicovitch*** și a fost calculată pe baza măsurătorilor de la aproximările simple ale unei curbe. Mai târziu, ***Benoit Mandelbrot*** i-a dat ***dimensiunii Haussdorff/Besicovitch*** numele de ***dimensiune fractală***, ea devenind astfel unealta care le-a permis matematicienilor să măsoare, din acel moment, un univers mult mai mare de forme, care înainte ar fi fost imposibil de măsurat. Astfel, dacă matematica și geografia convenționale, prin modul lor de abordare, ne asigură un ajutor insignifiant în analiza multora dintre caracteristicile naturale ale geografiei, geometria fractală vine în ajutorul nostru, ca unealtă eratică de studiere a formelor complexe din natură, forme care, deși nu sunt arii, linii sau puncte, se încadrează în tradiționalele categorii ale dimensiunilor topologice.

Cu ajutorul unei dimensiuni fractale D care va fi cuprinsă între 1 și 2, se poate descrie natura complexă a formelor geografice. Dimensiunea fractală permite folosirea unor valori reale. Dacă, în mod obișnuit, considerăm că un punct are dimensiunea topologică 0, o dreaptă are dimensiunea topologică 1, o suprafață are dimensiunea topologică 2, iar un corp solid are dimensiunea topologică 3, la fel de bine putem afirma (fără să greșim) că o curbă complexă poate schița o suprafață (în cazul extrem dimensiunea poate fi 2). Pentru o curbă, dimensiunea fractală este cuprinsă între 1 și 2, în funcție de complexitatea de care dă dovadă (dimensiunea fractală crește pe măsură ce crește complexitatea). Dimensiunea fratală va fi diferită în funcție de zona care se va analiza din cadrul unui anumit obiect, deci unele forme geografice pot fi extrem de complexe în timp ce alte porțiuni pot fi extrem de simple.

Astfel, dimensiunea fractală D prezintă un rezumat al naturii fractale a porțiunii analizate, putând fi utilizată drept index pentru măsurarea complexității unor curbe sau poligoane distincte și ca o măsură alternativă de măsurare a compactării.

În cazul utilizării pentru evaluarea complexității marginilor unei forme a dimensiunii fractale apar următoarele avantaje, care nu sunt întâlnite în cazul măsurătorilor convenționale a gradului de compactare:

- dimensiunea fractală este superioară în evaluarea întregii zone a complexității curbei, având abilitatea de a rezuma complexitatea acesteia.
- faptul că nu toate măsurătorile fractale necesită ca granița unui poligon să fie închisă, permite modificarea focalizării investigației și, în concluzie, granițele geografice care nu sunt subiect de studiu, pot fi înlăturate.

Analiza fractală asigură o apropiere mai puțin ambiguă în ceea ce privește analiza structurii și formei spațiale, decât măsurătorile de compactare.

Estimarea dimensiunii fractale

Există mai multe metode pentru estimarea dimensiunii fractale; în anul 1993 **Burrough** a exemplificat șapte astfel de metode pentru estimarea dimensiunii fractale D. Alți autori au exemplificat și ei astfel de metode: **Longley** și **Batty** în anul 1989, **Batty** în anul 1993 etc. Pentru respectivul experiment a fost selectată metoda numită *The Functional Box Count of a Curve* (**Burrough** - 1993), aceasta fiind utilizată pentru detectarea măririi lungimii granitei (curbei) unui district la niveluri din ce în ce mai fine ale detaliului. Acest lucru se realizează prin numărarea căsuțelor străbătute sau atinse de către graniță (curbă) în cadrul unui caroiaj. Numărul căsuțelor precum și lungimile unei părți a unei grile succesive sunt contorizate.

Când datele vor fi reprezentate pe un grafic logaritmic, rezultatele vor corespunde îndeaproape următoarei relații liniare (**Batty** - 1989, **DeCola** - 1993):

$$\log \textit{numărul\_de\_căsuțe} = a + b \log \textit{dimensiunea\_căsuței},$$

unde *numărul_de_căsuțe* reprezintă numărul căsuțelor străbătute, iar *dimensiunea_căsuței* reprezintă lungimea laturii unei căsuțe. Intersecția este reprezentată de a, iar b reprezintă panta. Valoarea absolută a pantei b reprezintă dimensiunea fractală D.

Când este realizată o regresie liniară pentru a potrivi o linie în setul actual de observații, iese la iveală următoarea relație:

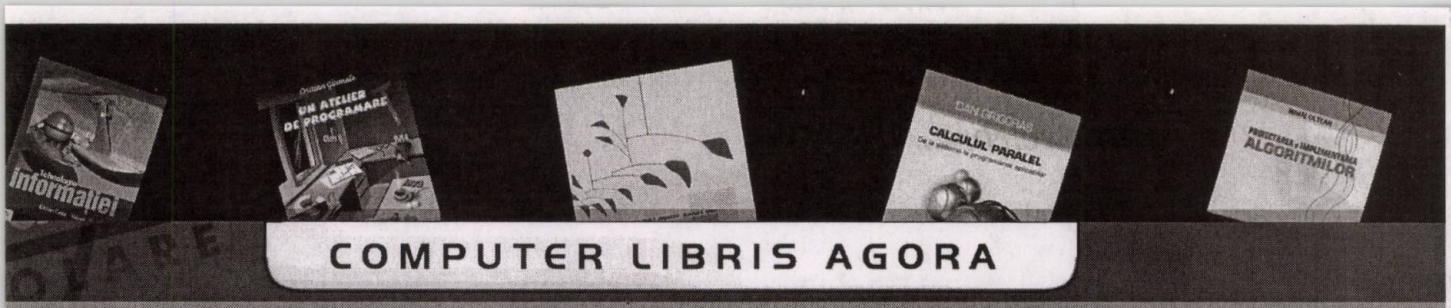
$$\log \textit{numărul\_de\_căsuțe} = 1.78 + 1.43 \log \textit{dimensiunea\_căsuței}$$

Dimensiunea fractală D este reprezentată de valoarea absolută a pantei b, deci putem spune că dimensiunea fractală pentru numărul funcțional de căsuțe ale granitei districtului este egală cu 1.43.

Bibliografie

1. Alain Boutot, *Inventarea formelor*, Editura Nemira, 1996
2. Dick Oliver, *Fractali*, Editura Teora, 1996
3. *Solstice an Electronic Journal of Geography and Mathematics*, Volume I, Number 1, Summer, 1990

Cristian Kalanyos este student în anul II la Universitatea Babeș-Bolyai din Cluj. Poate fi contactat prin e-mail la **kalanyosc17@yahoo.com**.

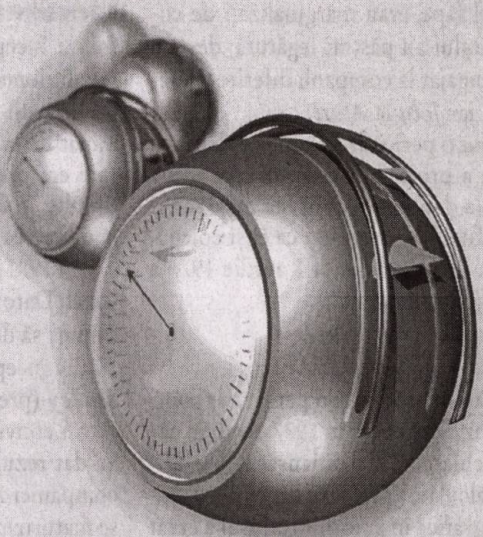


COMPUTER LIBRIS AGORA

DAN GRIGORAȘ

CALCULUL PARALEL

De la sisteme la programarea aplicațiilor



Calculul paralel. De la sisteme la aplicații tratează, într-un mod natural și structurat, cele mai importante aspecte ale proiectării aplicațiilor paralele, sau, cu un termen mai nou, de mare performanță. Paralelismul, care înseamnă executarea mai multor operații simultan, este o caracteristică a tuturor calculatoarelor actuale, inclusiv a celor personale. De aceea, cunoașterea principiilor și legilor care îl guvernează, ca și a metodologiilor pentru proiectarea aplicațiilor cele mai performante este o necesitate,dacă se dorește exploatarea optimă a resurselor sistemelor de calcul. Parametrii de performanță și metodologiile de proiectare și execuție optimă reprezintă subiectele care guvernează discursul științific al cărții de față.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

APPLE Computer

Claudiu Soroiu

Compania Apple a avut o influență semnificativă asupra industriei tehnologiei informației. În acest articol vom încerca să prezentăm un scurt istoric al acestei firme, punctând evenimentele importante care au avut loc de-a lungul timpului.

Nașterea companiei

Compania *Apple* a fost creată de doi prieteni: *Steven Wozniak* și *Steven Jobs*. Ei s-au cunoscut în timpul liceului și aveau preocupări comune, amândoi fiind interesați de electronică și, datorită acestui fapt, erau marginalizați de colegi. După terminarea liceului au păstrat legătura, deși au renunțat la studii și s-au angajat la companii diferite (*Wozniak* la *Hewlett-Packard*, iar *Jobs* la *Atari*).

În 1976, după ce a lucrat o perioadă de timp ca *designer* de calculatoare, *Wozniak* a proiectat produsul care urma să devină *Apple I*. *Jobs* și-a dat seama că există mari șanse ca acest produs să fie profitabil și a insistat ca el și colegul său să încerce să îl vândă. Prin urmare, la 1 aprilie 1976 a fost creată compania *Apple Computer*.

Primele produse

Apple I nu a fost considerat un produs competitiv, iar compania nu a început să se dezvolte decât în 1977 când, în cadrul unei expoziții de calculatoare a fost lansat *Apple II*.

Acesta a fost primul calculator personal care avea o carcasă din plastic și folosea grafică în culori. Produsul a creat o impresie foarte bună, iar comenzile primite de *Apple* au crescut de câteva ori imediat după lansare.

O nouă creștere semnificativă a vânzărilor a avut loc la începutul anului 1978 când a fost lansat produsul *Apple Disk II*. Această unitate pentru dischete era cea mai ieftină și cea mai ușor de folosit dintre toate produsele similare disponibile în acel moment pe piață.

Creșterea vânzărilor a dus la dezvoltarea companiei și, în 1980, când a fost lansat *Apple III*, numărul angajaților a ajuns la câteva mii și au început vânzările în străinătate. *Apple* a angajat mai mulți *manager*-i experimentați și au apărut câțiva noi investitori care au devenit membri ai Consiliului de Administrație. Noii directori, mai în vârstă și mai conservatori, au avut grijă ca *Apple* să devină o "companie adevărată", lucru care i-a îngrijorat pe cei care erau angajați de mai mult timp.

În anul 1981 lucrurile s-au complicat: pe o piață saturată era destul de dificil să se mai comercializeze calculatoare... Datorită acestui fapt, *Apple* a fost nevoită să concedieze 40 dintre angajați. În această perioadă *Wozniak* a fost rănit într-un accident de avion și *Jobs* a devenit *chairman* al companiei *Apple Computer*.

Plecarea lui Jobs

Jobs a început să dezvolte *Lisa*, un produs care urma să revoluționeze industria calculatoarelor. Din nefericire, el s-a dovedit a nu fi un bun *manager* de proiect și a fost eliminat de la conducerea acestuia de către *Mike Markkula*, care era președintele companiei și deținea majoritatea acțiunilor. *Jobs*, care deținea doar 11% din acțiuni, a preluat conducerea unui alt proiect: *Macintosh*.

În 1981 *IBM* a lansat pe piață primul său calculator personal. Datorită puterii financiare a acestei companii, *PC*-ul a ajuns să domine în scurt timp piața.

La începutul anului 1983, *Jobs* a insistat pe lângă *John Sculley* (președintele companiei *Pepsi-Cola*) în încercarea de a-l convinge să preia *Apple*. În aprilie, demersurile sale au dat rezultate și *Sculley* a devenit președinte și *CEO* al companiei *Apple*. *Jobs* credea că *Sculley* va ajuta *Apple* să se maturizeze, dar nu avea nici o idee despre întorsătura pe care o vor lua lucrurile.

Deși era un om de afaceri de succes, s-a dovedit destul de repede că *Sculley* nu știa prea multe despre industria calculatoarelor; imediat, s-au iscat certuri între el și *Jobs*.

Pe data de 22 ianuarie 1984, în timpul celei de-a treia reprize a *Super Bowl*-ului, *Apple* a lansat un spot publicitar de 60 de secunde în care prezenta produsul *Macintosh*. La început, vânzările au fost foarte mari, dar spre sfârșitul anului consumatorii nu au mai fost mulțumiți de acest produs, datorită cantității mici de memorie *RAM*.

Certurile dintre *Sculley* și *Jobs* s-au amplificat la începutul anului 1985. În luna mai *Jobs* a decis să încerce să preia compania. I-a stabilit lui *Sculley* o întâlnire de afaceri în *China* și a convocat Consiliul de Administrație în timp ce acesta era plecat. În ultimul moment *Sculley* a aflat de planul lui *Jobs* și a hotărât să se confrunte cu acesta. După o dispută aprigă între cei doi, Consiliul a votat în unanimitate în favoarea lui *Sculley*. *Jobs* a demisionat în aceeași zi, iar *Sculley* a rămas singur la conducerea companiei.

Perioada Sculley

John Sculley a devenit cu adevărat conducător al companiei în mai 1985. În următoarele câteva luni, a fost nevoit să concedieze o cincime dintre angajați. În scurt timp a fost anunțat primul trimestru financiar în care *Apple* a înregistrat pierderi.

În aceeași perioadă a început conflictul dintre *Sculley* și *Bill Gates*, datorită faptului că suprafața de operare *Microsoft Windows 1.0* avea o interfață grafică foarte asemănătoare cu cea a *Mac*-urilor. Până la urmă, *Gates* a fost convins să semneze un acord prin care *Windows 1.0* să nu folosească tehnologii *Mac*, dar în care nu se specifica nimic despre viitoarele versiuni *Windows*. Avocații *Microsoft* au avut grijă ca această companie să profite de situație și *Apple* și-a pierdut drepturile exclusive asupra *design*-ului său grafic.

Ieșirea din impas a fost datorată, în primul rând, lansării aproape simultane a produselor *LaserWriter* și *PageMaker*. Primul dintre aceste produse a fost prima imprimantă laser *PostScript* pentru *Mac*-uri, accesibilă ca preț, iar al doilea a fost unul dintre primele programe pentru tehnoredactare. Cele două lansări au făcut ca *Mac*-urile să devină soluția ideală pentru publicarea cu costuri reduse.

În 1987 a apărut *Mac II*, un calculator proiectat pentru a fi ușor expandabil. Acest produs a făcut ca *Macintosh* să devină o familie de calculatoare puternice și viabile.

În 1989 se părea că proiectul *Microsoft Windows* va fi un eșec și că *Mac*-urile vor domina și în următorul deceniu piața calculatoarelor. Nu a fost așa; în 1990 piața era saturată de clone *PC*, având cele mai diverse configurații, în timp ce *Apple* era singura companie care vindea *Mac*-uri.

În luna mai a apărut *Windows 3.0* care putea rula pe toate clonele *PC* disponibile pe piață. *Apple* se afla în dificultate.

Soluția imaginată de conducătorii companiei a fost de a licenția sistemul de operare *Mac OS*. A existat și propunerea de a porta sistemul de operare pe mașini care foloseau procesoare *Intel*. Ideea a fost abandonată atunci când *Michael Spindler*, unul dintre noii conducători ai companiei a afirmat că ***e prea târziu pentru licențe.***

La sfârșitul anului 1991 *Apple* a lansat prima generație de produse *PowerBook* care s-au bucurat imediat de un important succes pe piață. Se lucra la un nou tip de calculator: *PDA* (*Personal Digital Assistant*); *Apple* a dat noului produs denumirea *Newton*.

Sculley s-a arătat imediat interesat de acest produs și a grăbit lansarea, care a avut loc în august 1993. Prima generație de produse *Newton* folosea o tehnologie de recunoaștere a scrisului nu foarte performantă, iar volumul vânzărilor a fost sub așteptări.

Treptat, *Sculley* și-a pierdut interesul față de desfășurarea activităților companiei. Până la urmă, Consiliul de Administrație a decis că această stare de lucruri nu mai poate continua și i-a retras lui *Sculley* funcția de *CEO* al companiei. *Spindler* a devenit noul *CEO*, iar *Sculley* a rămas *chairman* pentru câteva luni după care a demisionat.

Decăderea

Spindler a fost, din aproape toate punctele de vedere, omul nepotrivit pentru funcția de *CEO* al companiei. Totuși, în cei doi ani și jumătate în care a condus *Apple*, au existat câteva realizări importante.

În 1994 a fost anunțată familia de produse *PowerMac*, primele calculatoare bazate pe cipul *PowerPC*, dezvoltat în colaborare cu *IBM* și *Motorola*. Procesorul *PowerPC* a permis calculatoarelor *PowerMac* să concureze cu calculatoarele care foloseau ultimele procesoare *Intel*. În multe cazuri produsele *Mac* erau mai performante decât cele oferite de concurență.

Spindler a reușit să vândă licențe *Mac OS* câtorva companii printre care se numără și *Power Computing*, unul dintre cei mai de succes producători de clone *Macintosh*. Mulți considerau că acordurile de licență erau prea restrictive, motiv pentru care doar câteva companii au cumpărat astfel de licențe.

Cea mai mare problemă cu care se confrunta *Apple* nu era vânzarea calculatoarelor, ci producerea lor. În iunie 1995 existau comenzi în valoare de un miliard de dolari, iar *Apple* nu avea la dispoziție componentele necesare. Problemele s-au agravat în momentul lansării sistemului de operare *Microsoft Windows 95* a cărui interfață grafică imita interfața *Mac* mai bine ca niciodată.

Cel mai important declin a avut loc în iarna 1995-96. Evaluând greșit cerințele pieței, *Apple* a favorizat produsele ieftine *Performas* față de *PowerMac*-uri. Astfel, *Apple* nu a înregistrat nici un profit și a încheiat trimestrul cu o pierdere de aproximativ 68 de milioane de dolari.

În ianuarie 1996 lui *Spindler* i s-a cerut să demisioneze din funcția de *CEO* și a fost înlocuit de *Gil Amelio*, fost președinte al companiei *National Semiconductor*.

Amelio a depus un efort susținut pentru ca *Apple* să fie din nou profitabilă, dar nu a avut succes. După 100 de zile de conducere, el a anunțat o mulțime de schimbări în structura companiei. Aceasta a fost împărțită în șapte departamente separate, fiecare dintre ele devenind responsabile pentru profiturile sau pierderile proprii.

Chiar dacă primul trimestru financiar al anului 1996 s-a încheiat cu pierderi în valoare de aproximativ 740 de milioane de dolari, în al doilea trimestru pierderile au fost de doar 33 de milioane de dolari, iar în al treilea trimestru compania a înregistrat profituri de 30 de milioane de dolari.

La sfârșitul anului, *Apple* a anunțat intenția de a achiziționa compania *NeXT* și revenirea lui *Steven Jobs* în cadrul companiei. Preluarea firmei *NeXT* a avut ca scop achiziționarea drepturilor asupra produsului *NeXTstep*, care va sta la baza următoarei generații a sistemului de operare *Apple*: *Rhapsody*.

Departamentul care se ocupa cu fabricarea produselor *Newton* a fost transformat în compania *Newton Inc.*, deținută integral de *Apple*.

În iulie 1997 *Gil Amelio* a demisionat în urma unui nou trimestru financiar încheiat cu pierderi. *Steven Jobs* a asigurat interimatul la conducerea companiei.

Revenirea

Prezența lui *Jobs* a fost simțită de îndată ce *NeXT* a fost achiziționat. Rolul lui *Jobs* a devenit evident. Fără *CEO* și cu cotația acțiunilor ajunsă la un nivel mai mic decât minimul ultimilor cinci ani s-a ajuns în situația în care trebuiau luate o mulțime de decizii și s-a intrat într-o criză acută de timp. *Jobs* a început să facă schimbări majore în structura companiei, cele mai multe anunțuri surprinzătoare fiind date publicității în august 1997 la expoziția *Mac World* din *Boston*.

Jobs, care era acum *CEO* interimar al companiei, a prezentat viitoarea campanie publicitară agresivă pentru viitoarele *Mac*-uri și sistemul de operare *Rhapsody*. Au fost anunțați și membri unui Consiliu de Administrație format aproape în totalitate din persoane noi. Printre aceștia se numără *Larry Allison*, care avea funcția de *CEO* la compania *Oracle*. Cea mai surprinzătoare știre a fost ultima: *Jobs* a anunțat un acord cu *Microsoft*.

Pentru a elimina diferite clone a produselor sale, *Apple* a început să răscumpere licențele *Mac OS* acordate diferitelor companii cum ar fi *Power Computing*, *Motorola* și *IBM*. La 10 noiembrie 1997 a avut loc o nouă conferință de presă în care *Jobs* a anunțat noi modificări în strategia companiei *Apple*. Se vor vinde calculatoare direct fie prin intermediul *Web*-ului, fie prin intermediul telefonului. Au fost anunțate și două calculatoare *Apple* noi: *PowerMac G3* și *PowerBook G3*. *Site*-ul *The Apple Store* a avut un succes extraordinar și într-o săptămână a ajuns cel de-al treilea *site eCommerce* din lume.

La expoziția *Mac World*, desfășurată în ianuarie 1998 la *San Francisco*, *Jobs* a anunțat că ultimul trimestru a fost, după mult timp, unul profitabil, câștigurile companiei ridicându-se la suma de 44 de milioane de dolari. În aprilie *Jobs* a anunțat un nou trimestru profitabil cu câștiguri de 57 de milioane de dolari. La începutul lunii mai a fost anunțat un nou *PowerBook G3*, *site*-ul *Educational Apple Store* și un produs cu un *design* complet nou: *iMac*. Tot în luna mai a fost anunțat sistemul de operare *Mac OS X*.

În iulie 1998 a fost anunțat un al treilea trimestru consecutiv profitabil, de această dată veniturile fiind de 101 milioane de dolari. Valoarea acțiunilor a crescut foarte mult, *iMac* a devenit cel mai bine vândut calculator din *SUA*, iar vânzările au depășit cele mai optimiste previziuni. A fost anunțat un nou trimestru profitabil și astfel s-a încheiat un întreg an în care compania *Apple* nu a mai avut pierderi. La începutul anului următor s-a încheiat al cincilea trimestru profitabil pentru *Mac*.

În iulie 1999 a fost introdus un calculator portabil numit *iBook*. Câteva luni mai târziu a fost anunțat *PowerMac G4*, un nou calculator de birou. Acțiunile *Apple* au crescut de-a lungul verii, iar la mijlocul lunii septembrie au ajuns la valoarea maximă din toate timpurile.

Ultima perioadă

În ianuarie 2000, la *Mac World Expo* din *San Francisco*, *Jobs* a prezentat noua strategie *Internet* a companiei: o suită de aplicații *Internet* disponibile doar în variante *Mac* și cunoscute sub numele de *iTools*. *Jobs* a anunțat intenția de elimina termenul "interimar" din titulatura funcției sale și a devenit *CEO* al companiei. Volumul și valoarea vânzărilor au continuat să crească, ceea ce a dus la o nouă creștere a valorii acțiunilor. În iulie 2000, *Apple* a anunțat o nouă familie de calculatoare numită *PowerMac G4 Cube*. Introducerea acestui produs a fost cel mai mare risc pe care și l-a asumat *Jobs* de la lansarea *iMac*. S-a dovedit că acest produs a fost un eșec total.

A doua jumătate a anului 2000 a fost o perioadă dificilă pentru *Apple*. Scăderea vânzărilor, combinată cu o înțelegere greșită a nevoilor consumatorilor a avut ca urmare un trimestru neprofitabil pentru *Apple*, după 3 ani. Unul dintre factorii acestui declin a fost *G4 Cube*, care s-a vândut într-un număr mic, în primul rând datorită prețului mare comparativ cu celelalte produse *Apple*. Un alt factor a fost decizia de a include *drive*-uri *DVD-ROM* în locul *drive*-urilor *CD-RW*. Astfel, *Apple* a ratat oportunitatea de a vinde produse clienților care doreau să aibă posibilitatea de a-și inscripționa propriile *CD*-uri. Rezolvarea acestor probleme a început la sfârșitul anilor 2000, când au fost reduse prețurile pentru întreaga linie de produse *PowerMac*. Următorul pas a fost realizat în ianuarie 2001, când *Apple* a introdus o nouă linie de produse *PowerMac* în care clienții puteau opta fie pentru o unitate *CD-RW*, fie pentru noua unitate *SuperDrive*, un dispozitiv capabil să citească și să inscripționeze atât *CD*-uri cât și *DVD*-uri. Au fost anunțate două noi aplicații: *iDVD* și *iTunes*. Aceasta din urmă permiteau utilizatorilor să creeze și să asculte melodii în format *MP3* și apoi să le inscripționeze pe *CD*-uri.

Toate acestea făceau parte dintr-o nouă strategie a companiei, dezvoltată datorită importantului regres al industriei tehnologice.

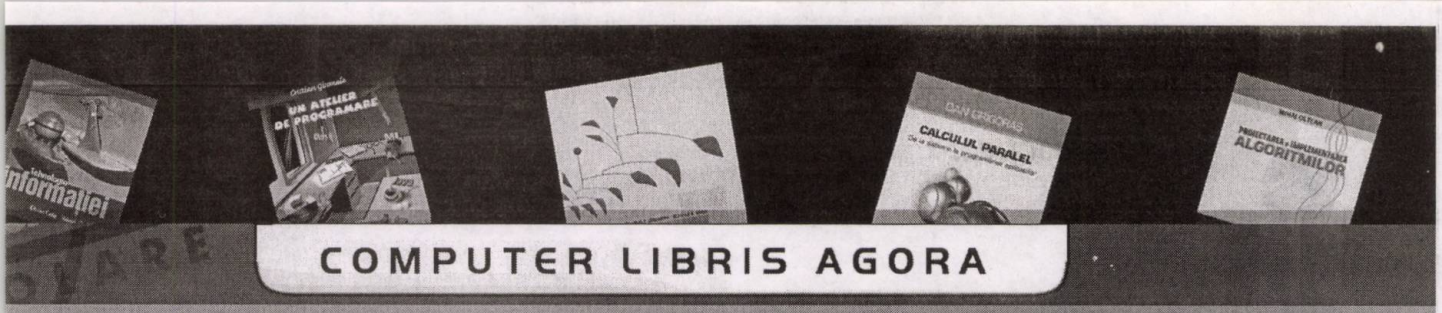
În mai 2001, *Jobs* a anunțat intenția companiei de a deschide un lanț de magazine pe întreg teritoriul *Statelor Unite*, în care nu se vindeau numai produse *Apple*, ci și alte dispozitive cum ar fi: *player*-e *MP3*, camere video și digitale, *PDA*-uri etc. Linia *iBook* a fost modificată, produsele împrumutând o mare parte a *design*-ului *PowerBook G4*. În iulie au fost actualizate liniile de produse *iMac* și *G4* și a fost suspendată producția pentru *G4 Cube*.

În toamna anului 2001 au fost revizuite liniile *PowerBook G4* și *iBook*, produsele *iBook* fiind foarte bine vândute în timpul verii. La sfârșitul lunii octombrie *Apple* a anunțat primul său produs din ultimii ani care nu era un calculator. Acesta se numește *iPod* și este un mic *player* digital de muzică, ce include și un disc hard. Acest produs se confruntă cu aceleași dificultăți ca și *G4 Cube*: a fost favorizat stilul și nu prețul. Este ultimul risc important asumat de *Apple*.

Bibliografie

- www.apple-history.com

Claudiu Soroiu este redactor GInfo și poate fi contactat prin e-mail la adresa csoroiu@yahoo.com.



COMPUTER LIBRIS AGORA

Cartea este o culegere de studii de caz, construite pe temelia problemelor concursurilor Southeastern European Regional Programming Contest din perioada 1995-1999, concursuri organizate sub egida Association for Computing Machinery.

Cartea se adresează atât începătorilor, cât și celor cu experiență în domeniul programării, în încercarea de a contura prin exemple imaginea unei lumi diverse și pasionante, dar pretențioase, aceea a înzestrării mașinii de calcul cu capacitatea de a rezolva corect și eficient.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

La închiderea EDIȚIEI

Carmen Șușman

Dragi cititori, să nu credeți că a edita o revistă este un lucru ușor... Cu toate acestea, fiind o redacție destul de tânără, am învățat destul de mult unii de la alții ce înseamnă să faci o revistă, sperăm noi, competitivă. La noi în redacție există și zile pline, aproape 12 ore de muncă, dar se întâmplă lucruri care ne fac să uităm cât suntem de obosiți...

Nu cred că mai are rost să vă fac cunoștință cu membrii redacției pentru că ne cunoașteți deja din caseta de la prima pagină. Așadar, voi trece direct la redarea faptelor:

Într-o zi intră *Mihai* în redacție:

- Salutare! Ce avem de făcut azi?

Eu tac. Îl las pe *Claudiu* să răspundă:

- Azi trebuie să trimitem *GInfo* la tipografie!

Mihai intră în panică. Se albește la față.

- *Claudiu*, ai terminat toate articolele de tehnoredactat?

- Nu.

- Ai scris articolele tale?

- Primele șapte rânduri dintr-unul...

- *Carmen*, ai corectat toate articolele?

- Da. Nu! Nu le-a listat Claudiu! Nu sunt publicabile toate!

- La treabă copilași! *Claudiu*, nu mai sta pe *Internet* că îți tai din salar! Scrie-ți articolele! Repede până nu mă enervez! Lasă **trafic.ro**-ul că nu ni se modifică poziția în câteva ore.

- *Carmen*, ți-am listat ultima variantă a articolului cu fractalii. Corecteaz-o repede!

După un timp mă ridic nervoasă de la birou. Am citit prima parte din articol și am observat că autorul nu a efectuat nici o corectură făcută de mine pe prima variantă trimisă de el.

- Ce faci *Claudiu*? întreabă *Mihai* speriat văzând o minunăție de ecran albastru.

- Resetez calculatorul! S-a blocat sistemul!

- Lucram pe discul tău și nu am salvat. Trebuia să mă anunți că resetezi, că aș fi salvat...

- Scuze!

- Ce scuze!? Acum să faci tu și asta că m-am enervat!

- Ok!

- Ok, ok! Știu că tot eu va trebui să o fac până la urmă!

În redacție intră doamna *Clara*, care trebuia să aibă pe birou toată gazeta listată pentru a o corecta... La sosirea redactorului-șef se face liniște în redacție. *Mihai* o întâmpină zâmbitor:

- Bună ziua! Doamna *Clara*, vreți să vă spun un cuvânt magic?

- Da! răspunde doamna *Clara* plină de entuziasm.

- "Editorial!"

Doamna *Clara* devine agitată. Zâmbetul dispare! Întreabă dacă avem ceva idei; noi începem să spunem cam tot ce ne trece prin minte, iar cel ghinionist este *Mihai*. De ce? Pentru că doamnei *Clara* i-a plăcut ideea lui și l-a însărcinat pe el să scrie faimosul editorial!

- Când trebuie trimisă gazeta? întreabă doamna *Clara*.

Tăcere. Nimeni nu mai zice nimic. *Claudiu* se oprește din butonat, *Mihai* privește pe fereastră, iar eu... eu mă fac mică, mică de tot.

- Hai, copilașilor! Răspundeți-mi!

- Azi, zice *Mihai* până la urmă.

Doamna *Clara* se gândește, nu prea știu eu exact la ce, dar imediat vine întrebarea de care mi-era mie frică.

- Și cât la sută din gazetă e gata?

- 25% plus 75% astăzi!

Din acel moment în redacție se face liniște. Toată lumea se concentrează asupra sarcinilor pe care le are de îndeplinit. Dacă sună telefonul tresărim toți, dacă cineva vorbește imediat i se dă replica:

- Taci! Pune mâna și lucrează!

După câteva ore, *Claudiu* sare de pe scaun!

- Am uitat să merg la laborator! Mi-am făcut tema degeaba azi-noapte! Din cauza voastră am uitat! M-a pus absent! În loc să merg la școală am stat aici să mă chinui cu ecranele albastre.

- Lasă, mergi săptămâna viitoare! Stai jos și continuă ce făceai! Trebuie să terminăm revista astăzi!

Gata, l-am calmat, îmi zic eu mulțumită. Cred că ați observat că toți trei suntem studenți.

Când îmi ridic ochii din foaie îl văd pe *Claudiu* jucându-se de zor *FreeCell*! Mă enervez!

- Noi lucrăm și tu te joci! Trădătorule! Îi zic lui *Mihai* să-i zică doamnei *Clara* să-ți taie din salar. Apoi, după câteva minute merg și eu ca să-l ajut, poate e un joc greu...

Tresărim atunci când se aude glasul lui *Mihai*.

- V-ați hotărât care e *site*-ul lunii?

Răspundem simultan:

- **www.haz.ro**, spun eu.

- **www.price.ro**, spune *Claudiu*.

Uitați-vă la pagina 8 dacă vreți să știți cu cine a fost de acord *Mihai*. Poate am mai mult noroc luna viitoare...

Era rândul lui *Mihai* să ne spună care este jocul lunii. Îmi plăceau mai multe jocuri care apăruseră de curând, dar știam că domnul redactor-șef adjunct este pasionat de *Black and White*. Și cum tocmai fusese lansat *add-on*-ul pe care îl aștepta atât de mult, nu prea avea rost să încerc să-i dau idei. Îmi aduc aminte cum am râs toți de el când ne-a spus că a dat un milion pe licența jocului. Poate că ar trebui să faceți și voi la fel. Să nu-mi spuneți că nu vă jucați! Și firmele de jocuri trebuie să trăiască din ceva. Poate atunci când veți avea un salar comparabil cu al lui...

Avem noroc că la noi în redacție nu există discuții de genul: "*procesoarele AMD sunt mai bune!*", "*Intel bate AMD la orice oră!*", "*Windows-ul e superior Linux-ului din orice punct de vedere!*", "*Ba nu! Linux-ul e mai bun...*", "*Produsele Microsoft sunt pline de bug-uri!*". Vă dați seama ce certuri ar fi fost în momentul în care ar fi trebuit să alegem produsele *software* și *hardware* ale lunii.

Rubrica ***Actual*** este întotdeauna ultima la care lucrăm fiindcă vrem ca alegerile noastre să fie produse apărute într-un moment de timp cât mai apropiat de data apariției revistei. Având în vedere că discutăm despre ea, înseamnă că ne apropiem de final. Revista va fi gata peste câteva ore.

Între timp doamna *Clara* butona ceva... După un timp ne spune zâmbitoare că a terminat editorialul. *Mihai* a scăpat de data aceasta. Răsuflă ușurat. Editorialul ne place tuturor fiindcă nu vrem să mai stăm câteva ore până se mai scrie unul.

Și cum vedem noi că nu mai avem mult de lucru, ne gândim că ar fi timpul să luăm o pauză.

Credeți că la noi nu se chiulește? Da' nu!? Mai zicem câte unul:

- Merg să-mi cumpăr ceva de mâncare, vrea cineva ceva? Și pe cuvânt că se adună atât de multe chestii care trebuie cumpărate, încât musai să mergem doi, sau poate toți trei! După vreo 30-40 de minute ne întoarcem de la suc ăăă... , de la cofetărie ăăă... , de la cumpărături. Mda, era să mă dau de gol!

Evident că spre seară sună cei de la *Târgu-Mureș*.

- Alo, nu trimiteți revista odată, trebuie dusă la tipografie!

- Ba da, dar mai târziu puțin. Mai trebuie să creăm arhiva și să o trimitem.

- Și cam cât durează ca să trimiteți o arhivă?!

- Păi, o primiți peste câteva ore.

- Cum? Ați zis că revista e gata?

- Nu reușim să ne conectăm la *Internet*! Ne-ați lăsat conturile astea de *dial-up*! Cum credeți că putem să lucrăm în asemenea condiții?

Nici ei nu mai au chef de încă o ceartă referitoare la conexiunile *Internet*, așa că renunță. Vocea de la telefon îmi spune:

- Bine!

Mă întorc triumfătoare de la telefon.

- Băieți! Pauză! Nu au zis că e urgent, deci o putem trimite și mâine seară, așa că eu merg la școală că am un curs și pe urmă vin înapoi.

Dar n-am apucat să plec pentru că telefonul a sunat iar, și iar și o altă vocea spunea ceva de genul:

- Reviiistaaa!!!

sau

- Gazzzeeetaaa!!!

La două și jumătate noaptea oprim calculatoarele. Suntem frânți de oboseală. Vă puteți închipui că abia acum putem să răsuflăm ușurați. Arhiva plecase, nimic nu mai depindea de noi. Ne hotărâsem chiar și ce va apărea în numărul următor. Eram mândrii de noi. Dar eu pun o simplă întrebare:

- *Claudiu*, ai scris ianuarie 2002 în loc de decembrie 2001? Sper că...

- Am uitat! Stați să le trimit un *mail* celor de la *Târgu-Mureș* și să-i rog să schimbe ei...

După zece minute în care a resetat calculatorul de vreo cinci ori, Claudiu ne spune:

- O.K. Am rezolvat! Sper...

- Haideți să mâncăm ceva.

- Unde? Toate sunt închise la ora asta!

- Bine! Jucăm avioane? O tură, pe urmă mergem acasă. Mie mi-a sărit somnul!

- O.K.

Și începem să răscolim redacția pentru a găsi foi cu pătrățele. Ei da, suntem și noi oameni!... normali...

Ca de obicei, *Claudiu* câștigă. Eu și *Mihai* ne enervăm și cerem revanșa. Mai câștig și eu câte un joc, mai are și *Mihai* noroc câteodată și partidele continuă.

Suntem surprinși de pașii care se aud apropiindu-se. Intră doamna *Clara*, zâmbind.

- Așa de repede ați venit astăzi? Mai aveți de lucru la revista? Pot să vă ajut cu ceva?

- Am terminat acum vreo două ore!

- Ceeeeeeee? Ați stat toată noaptea aici! *Mihai*, ai zis că azi ai examen.

- Am, dar mai târziu... Nu vreau să mă gândesc la examene acuma. Îmi trebuie doar o cafea!

- Cine merge după cafea?

- Tragem la sorți!

Pe nesimțite, intră domnul *Eugen* și spune:

- Am tras eu la sorți. *Claudiu*, a ieșit că trebuie să mergi după cafea!

În numărul URMĂTOR

La mijlocul lunii februarie va apărea numărul 12/2 (februarie 2002) al revistei noastre. Vă prezentăm pe scurt câteva dintre articolele pe care le veți putea citi...

Un nou GeForce

Viitoarea generație a chip-urilor video produse de compania nVidia va fi lansată la începutul lunii februarie. Probabil, denumirea noului produs va fi GeForce 4. Va exista și o variantă mai ieftină a produsului care va purta denumirea GeForce 4 MX. Potrivit ultimelor informații date publicității, varianta destinată calculatoarelor performante va folosi sufixul Titanium. În numărul următor vom realiza o scurtă prezentare a noutăților introduse de noile produse.

Bursele Agora

Ultima etapă a concursului de programare organizat de GInfo va începe pe data de 1 martie 2002. Cei care doresc să afle mai repede enunțurile celor trei probleme le vor putea găsi în revistă. De asemenea, în numărul următor vor fi prezentate soluțiile problemelor de la a doua rundă, clasamentul rundei, precum și clasamentul intermediar după două etape. Soluțiile sunt disponibile pe site-ul oficial al concursului începând cu data de 15 Ianuarie 2002. Tot pe site veți putea găsi și fișierele de intrare folosite pentru a testa programele concurenților, împreună cu o variantă corectă a fișierelor de ieșire.

Tehnici de criptare

Codificarea informațiilor a fost un domeniu de interes încă din cele mai vechi timpuri. În numărul următor vom începe o serie de articole în care vom prezenta diferite tehnici de criptare. Vom începe cu algoritmii simplii folosiți în antichitate pentru a transmite informații secrete fără ca inamicii să reușească să descifreze textele trimise și vom continua cu diferiți alți algoritmi folosiți de-a lungul timpului până vom ajunge la ultimele noutăți în domeniu.

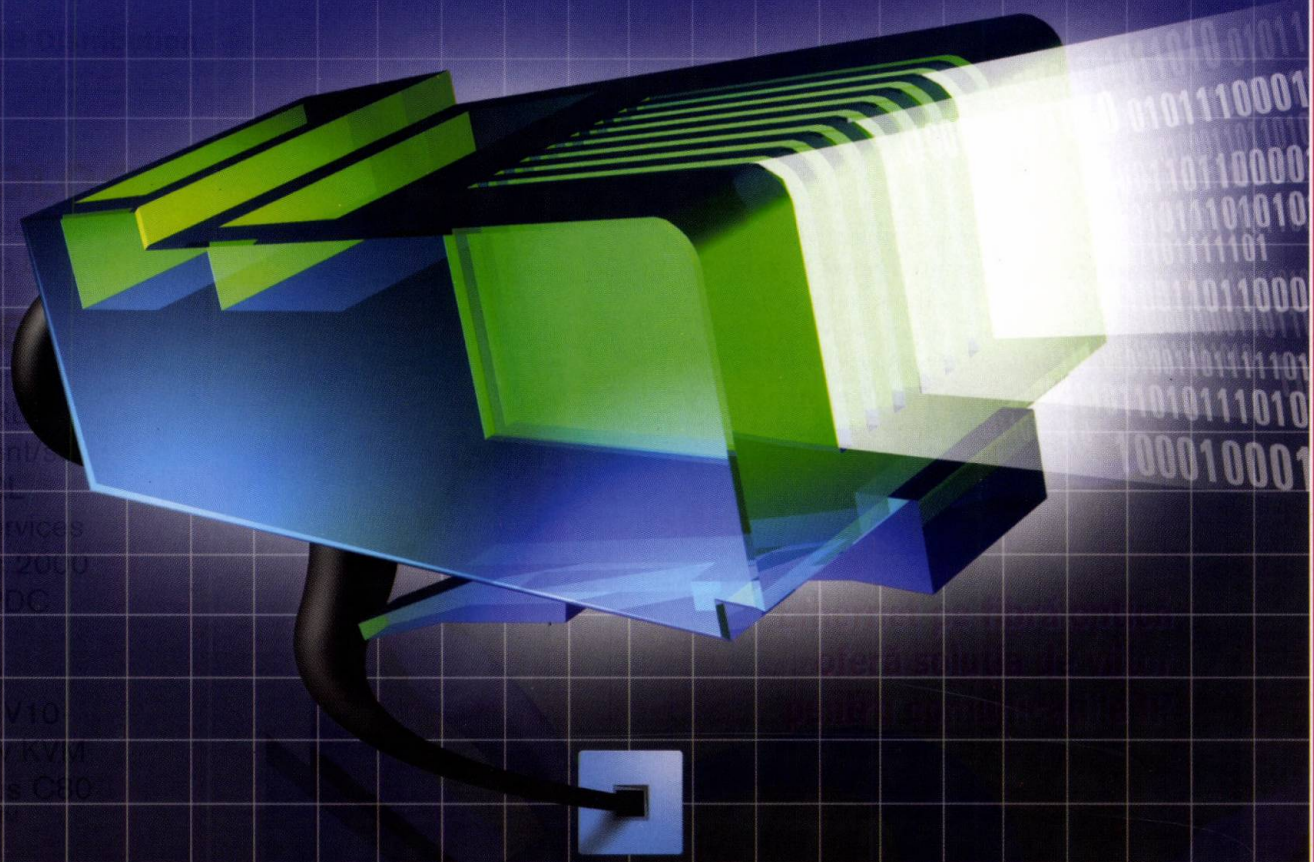
Intel

Cu siguranță, compania Intel este una dintre cele mai cunoscute firme din domeniul tehnologiei informației. Vă vom prezenta în numărul următor istoria acestei companii începând cu crearea acesteia și până în prezent. Veți putea afla detalii despre produsele lansate pe piață de companie de-a lungul timpului.

Războiul browser-elor

Cu siguranță, cele mai cunoscute programe de navigare sunt Internet Explorer și Netscape Navigator. În ultimii ani a existat o concurență acerbă între companiile Microsoft și Netscape care dorea, fiecare, să impună pe piață propriul produs. Bătălia a fost câștigată, fără drept de apel, de către Internet Explorer, browser folosit în prezent de peste 80% dintre cei care navighează pe Internet. Victoria a fost datorată, în primul rând includerii produsului în cadrul sistemului de operare Windows. Deși cu pași timizi, recent a apărut un nou concurent și anume Opera. Acum, toate cele trei browser-e au ajuns la versiunea 6. Vom realiza o prezentare comparativă a celor trei rivali, urmând ca dumneavoastră să decideți care dintre aceste produse este mai adecvat pentru activitatea pe care o desfășurați.

NETREPORT



Conectat la viitorul digital

AGORA MEDIA: Str. Tudor Vladimirescu nr. 63/9, Tg. Mureș, CP 230/1, 4300
Tel: 065-166 516, Fax: 065-166 290, e-mail: office@agora.ro

Agora MEDIA

Publicarea acestei reviste este posibilă datorită sponsorizărilor acordate de următoarele firme care s-au decis să sprijine învățământul românesc de informatică

SPONSORI DE PLATINĂ

COMPAQ

SPONSORI DE AUR

FORTE COMPUTERS
Flexibility. Made in Romania

ASTRAL TELECOM

SPONSORI DE ARGINT

sat
IT SOLUTIONS & SERVICES

hp
invent

ERICSSON

Magua

M.G.T.
EDUCATIONAL

a&C INTERNATIONAL S.A.
the CAD-CAMpany
autodesk
authorized distributor

ALCATEL

*Rugăm companiile care doresc să se alăture acestui efort să ne contacteze prin e-mail la adresa office@ginfo.ro sau la numărul de telefon: 065-166.516

Gifo
Februarie 2002
Vol. 12/2
Lei 20.000

gazeta de informatică

Revista viitorilor specialiști în tehnologia informației

din sumarul numărului:

Internet

OPERA software

RĂZBOIUL browserelor

Perpetuum mobile • Tehnici de criptare • Intel Corporation
Introducere în procesarea imaginilor • Star Trek • GeForce4
Concurs - partea I - Introducere, Soluții partea II

GHIDUL TĂU ÎN LUMEA TEHNOLOGIILOR

Misiunea noastră este de a testa și evalua produse și servicii din domeniul tehnologiei informației și comunicațiilor și de a prezenta rezultatele cu onestitate și obiectivitate

AGORA MEDIA Str. C-tin Rădulescu Motru nr. 13, etaj 4, apt. 42, sect. 4 (CP 94/49), București
Tel: 01-3309282, Fax: 01-3309285

www.pcmagazine.ro

Agora MEDIA

Vol. 12, Nr. 2 /2002

gazeta de informatică

Redacția:

Director științific:
prof. dr. *Horia Georgescu*
Redactor șef:
lect. univ. *Clara Ionescu*
Redactor șef adjunct:
Mihai Scorțaru
Redactori:
Carmen Șușman
Claudiu Soroiu
Au mai colaborat:
Cristian Kalanyos (Cluj)
Ciprian Ileană (Alba Iulia)
Mihai Stroe (București)
Tehnoredactare:
Claudiu Soroiu
Coperta și grafica:
Andrei Pașa
Adresa pentru corespondență:
Computer Libris Agora,
Universității 8/8,
3400, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 064/192.422
065/166.516
Fax: 065/166.290
http://www.ginfo.ro
ISSN: 1222-7129

Tiparul: *Master Druck* Târgu Mureș
tel: 065/162.359

Editor: Agora Media SRL
Agora Media SRL este membru al Biroului Român pentru Auditarea Tirajelor
Editor al publicațiilor "eWeek România", "PC Magazine România", "NetReport" și "Gazeta de Informatică"
str. T. Vladimirescu nr. 63/9
4300 Târgu-Mureș

Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director tehnic:
Szabó László
Vânzare reclamă:
Mihai Bucuroiu
Bogdan Manu
Florin Tiță
Denis Marioara
Marius Comărniceanu
Reprezentanța București:
Director: *Dana Pastramă*
Tel: 01/330.92.81
Fax: 01/330.92.85

Σ Sumar

actual.. PAG. 5

Produsele lunii
GeForce 4
Borland C++ BUILDER 6
Bridge Commander
Cine știe, CÂȘTIGĂ

concurs.. PAG. 9

Rezolvările problemelor celei de-a doua runde

Clasamentul general

Problemele ultimei runde a ediției 2001/2002

focus.. PAG. 18

Modelarea EXPRESIILOR folosind tehnici EVOLUTIVE

Mihai OLTEAN
Crina GROȘAN

~Termeni, factori~

tehno.. PAG. 23

MODELE pentru INTERCONECTARE

Mihai SCORȚARU

~Rețele de calculatoare~

serial.. PAG. 30

Tehnici de criptare

Claudiu SOROIU

~primul episod~

babel.. PAG. 33

Introducere în PROCESAREA imaginilor

Radu-Daniel VATAVU

~paradisul pixelilor~

internet.. PAG. 40

Generația a ȘASEA

Mihai SCORȚARU

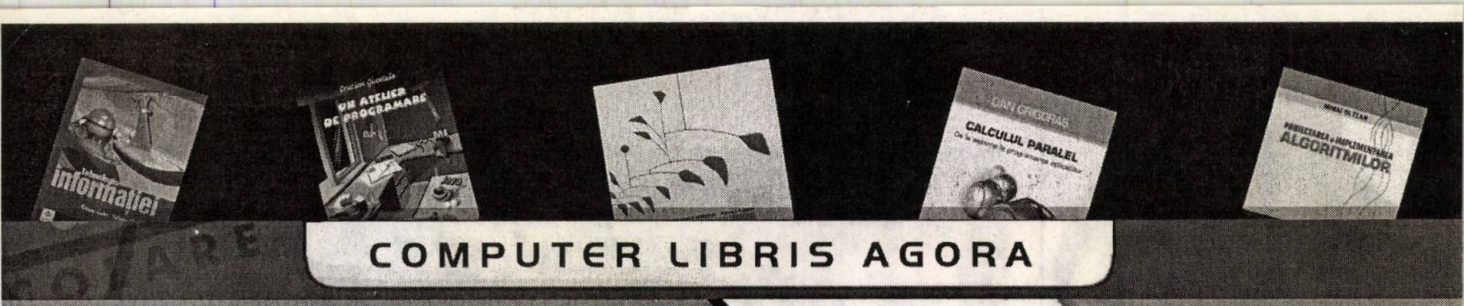
~Internet Explorer 6.0~
~Netscape Navigator 6.21~
~Opera 6.01~

profil.. PAG. 44

INTEL Corporation

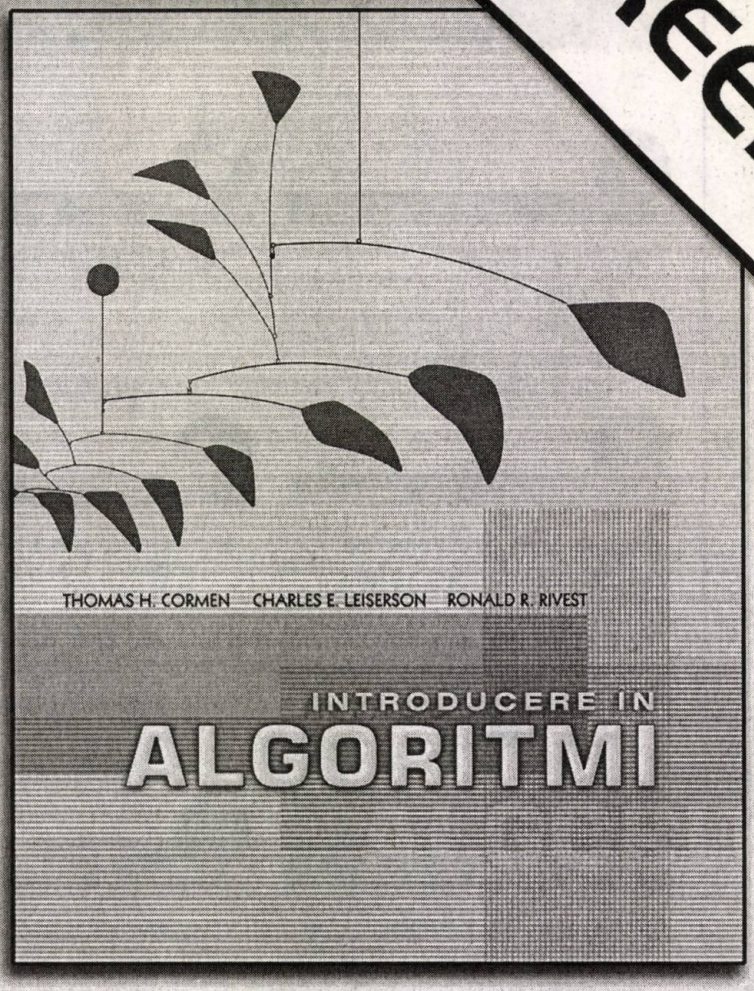
Carmen ȘUSMAN

~istoria unei firme celebre~
~istoria procesoarelor~



COMPUTER LIBRIS AGORA

REEDITARE



Cartea se adresează, în primul rând, celor care studiază la nivel elementar sau academic algoritmi și structuri de date. Deoarece tratează aspecte teoretice și practice, legate de proiectarea algoritmilor, este la fel de binevenită elevilor, studenților, profesorilor și autodidacților.
Introducere în algoritmi acoperă atât metodele clasice cât și metodele moderne ca analiza amortizată și algoritmii paraleli.
Prezentarea matematică, riguroasă, este atent detaliată în așa fel încât să fie accesibilă tuturor cititorilor.
Capitolele încep cu elemente de bază, avansând spre subiecte mai complexe.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

Perpetuum MOBILE

Cred că nu sunt singurul cadru didactic căruia i s-a replicat vreodată: "*Îți este ușor, ești profesor, numa' te duci în clasă și scrii pe tablă! Bașca vacanțele...*" Da, mulți ne văd așa. Dar cine ne-a invidiat pe vremea când, în loc să mergem la ștrand sau la cinema, "mucegăiam" prin biblioteci, aplecați asupra notițelor și cărților? Și nu este vorba doar de atât! În timpul multelor ore în care ne pregătim lecțiile și problemele propuse, ne văd doar cei apropiați și tot ei ne simt lipsa când stăm la școală lângă calculator, rezolvând probleme... Este evident că toți profesorii (indiferent de disciplina pe care o predau) se pregătesc pentru ore, toți consumă mult timp corectând lucrări, chiar dacă există diferențe privind cantitatea timpului și efortul intelectual depus de fiecare.

Și totuși, profesorii care predau informatica sunt "favorizați" de soartă... Ei învață mereu... E o bucurie, nu? Mereu apar lucruri noi, mereu se schimbă câte ceva și exact în cazul acestei discipline, în mod "miraculos" de multe ori elevii află noutăți mai repede decât profesorii. Și vin la școală și pun întrebări. "Știți, am văzut pe *Discovery...*", sau "am găsit pe *Internet*" sau... Bietul profesor de informatică se prinde de urechi și caută și el, se interesează, citește, cumpără cărți care costă jumătate din salarul lui pe o lună ca să poată ține pasul cu noutățile apărute. Așa-i trebuie, dacă s-a făcut prof de info...

Lăsând la o parte lamentările, aș vrea să semnalez un fapt de altfel cunoscut de către toată lumea: din nou se vor schimba programele școlare! Și știm ce înseamnă aceste schimbări. Putem să punem notițele folosite până acum în sertare și să gândim altele, să căutăm bibliografia corespunzătoare etc., într-un cuvânt: să o luăm de la capăt. Nu știu cât de normal sau anormal este ca programele școlare să se modifice atât de frecvent ca în România, dar de data aceasta se vor modifica din nou, nu doar la informatică, ci la toate disciplinele, din cauză că, începând cu anul școlar 2002/2003, clasa a IX-a trece în gimnaziu.

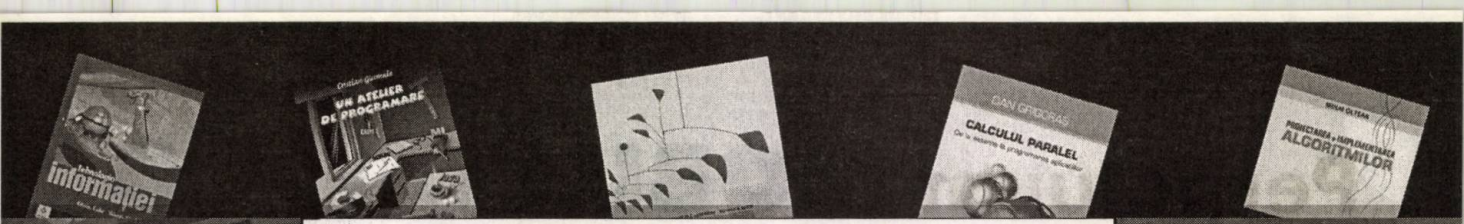
Oare ce se va întâmpla, mai ales cu acei copii care, pentru un singur an, vor fi smulși din școala frecventată până acum (unde au învățat sau nu...) și vor fi trimiși undeva, unde se va găsi loc pentru ei. Mi se pare absurd ca problema principală să fie *locul* și nu *copilul*... Dar să presupunem că suntem într-o astfel de clasă cu 25-30 de elevi care așteaptă ceva de la școală, de la profesori chiar și în acest an. Altfel ar trebui să acceptăm că vin absolut degeaba. Deci, se pune întrebarea: "*Ce vor învăța ei?*" Există posibilitatea să se "alungească" conținutul învățării în gimnaziu, se pot prevedea anumite cunoștințe din cele care erau studiate (și) până acum în clasa a IX-a, și evident se poate crea ceva special pentru acest an de studiu. Prima soluție n-o văd justificată, deoarece nu s-a pus problema *descongestionării* disciplinelor de gimnaziu. A doua nu ar putea să funcționeze eficient, din cauza rupturii cu mediul (cu fostul profesor de a IX-a, cu concepțiile fundamentale dintr-o anumită școală etc.). Rămâne a treia variantă sau o combinație a celor trei. Încurcată situație... Dar să ne oprim asupra disciplinei *Informatică* și să considerăm că în cadrul ei se învață utilizarea calculatorului în cele mai diverse scopuri. În foarte multe școli din România, unde există calculatoare, la orele de *Tehnologie*, în cadrul modulului *Informatică*, se învață cele mai diverse capitole ale acestei științe. Din păcate, aceste studii se derulează foarte variat, sunt structurate haotic, astfel încât este imposibil ca aceste cunoștințe să se constituie într-o bază temeinică. Unii profesori încep predarea *limbajului C* din clasa a V-a și încurajați de rezultate susțin că au perfectă dreptate. Alții îi lasă (obligă) pe copii să "butoneze" în *Word* ani la rând. Sunt școli unde predarea limbajului de programare *Logo* are o tradiție și copiii îl învață cu plăcere. În multe locuri se lucrează în *Basic*, în altele în *Pascal*. Problema nici nu ar fi așa de mare dacă, pe parcursul anilor din gimnaziu, pentru aceste studii s-ar găsi aplicații adecvate vârstei și, bineînțeles, cunoștințelor acumulate la alte discipline.

Din nou pun frâu comentariilor, pentru că mi se pare trist că unii profesori nu se gândesc la scopul predării; *ei pur și simplu predau ce știu*.

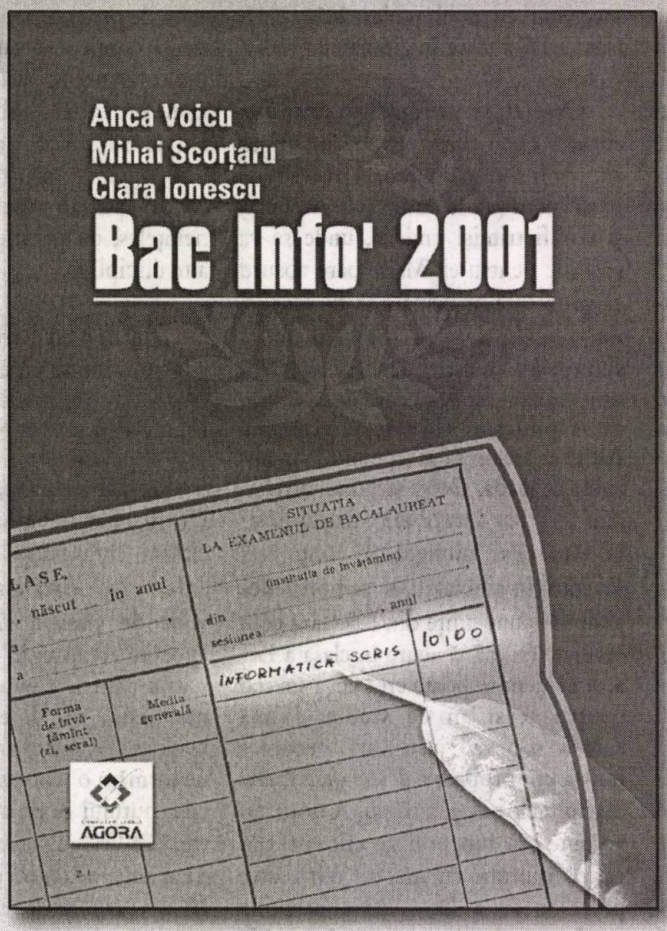
Ce părere aveți, dragi cititori ai *GInfo*, ce ar trebui să se predea la disciplina *Informatică* (sau *Tehnologia Informației*, dacă așa se va numi) elevilor de clasa a IX-a? Deschidem "porțile" revistei în care vom rezerva spațiu pentru a publica opinia dumneavoastră. Ar fi extraordinar dacă ne-ați scrie, astfel am putea și noi să ne formăm o viziune mai clară asupra conținutului dorit al... revistei noastre. Cât despre părerile exprimate până acum, voi aminti câteva: "*în clasa a IX-a să se facă doar teorie, asigurând cunoștințele necesare unei învățări practice în liceu*" (nu-i sigur că vor fi calculatoare peste tot...), "*să se învețe ceva care nu se bazează pe cunoștințe acumulate (sau nu) în clasele V-VIII, și pe care nu se clădește nimic în viitor, de exemplu HTML și JavaScript*". Bineînțeles există și alte opinii.

Închei cu speranța că această programă va ține cont de obiectivul principal al disciplinei: *să-i apropiem pe copii de informatică, de calculator, nu să-i îndepărtăm. Este în joc viitorul lor, modul în care își vor găsi drumul potrivit în viață.*

Clara Ionescu



COMPUTER LIBRIS AGORA



Bac Info' 2001 conține 20 de variante de subiecte pentru Bacalaureat. Fiecare variantă conține câte patru subiecte. Primul dintre ele conține trei subpuncte care testează capacitățile candidaților de a recunoaște valoarea de adevăr a unei propoziții, de a determina efectul unui anumit algoritm și de a găsi o anumită greșeală într-un algoritm. Al doilea subiect conține doua subpuncte; primul testează deprinderea candidaților de a folosi limbajul pseudocod, iar al doilea cea de a folosi un limbaj de programare (Pascal sauC++). Al treilea subiect este o problemă pentru rezolvarea căreia este necesară folosirea unor structuri dinamice de date, iar al patrulea este o problemă pentru a cărei rezolvare trebuie folosită metoda backtracking.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

GeForce 4

Compania nVidia a lansat la începutul acestei luni un nou chip pentru plăci video: GeForce 4. Acesta este cel mai performant produs de acest tip disponibil în acest moment pe piață, motiv pentru care a devenit produsul lunii februarie.

Introducere

Pe data de 6 februarie 2002 a fost lansată o nouă familie de cipuri grafice *nVidia*. Denumirea generică a acestora este *GeForce* 4, dar există trei tipuri de produse care fac parte din această familie: *GeForce* 4 *Ti* (pentru sistemele performante), *GeForce* 4 *MX* (pentru sistemele ieftine) și *GeForce* 4 *Go* (pentru calculatoarele portabile).

Date tehnice

Cipurile *GeForce* 4 folosesc arhitectura *LMA* II (*Lightspeed Memory Architecture*), o interfață pentru memoria *DDR* pe 128 de biți. Aceasta permite creșterea semnificativă a performanțelor memoriei.

Noile plăci grafice folosesc o tehnică numită ***Z-occlusion culling*** cu ajutorul căreia se poate determina mai eficient care dintre obiectele unei scene sunt vizibile și care obiecte sunt "mascate" de alte obiecte.

De asemenea, sunt folosite patru memorii *cache* dedicate operațiilor grafice și optimizate pentru efectuarea acestora. Acest sistem de memorii *cache* poartă denumirea de ***Quad Cache***.

GeForce 4 Ti

Familia plăcilor grafice *GeForce* 4 destinate calculatoarelor performante cuprinde două modele: *GeForce* 4 *Ti* 4400 și *GeForce* 4 *Ti* 4600.

- Plăcile *Ti* 4400 pot desena 125 de milioane de triunghiuri pe secundă, pot efectua 1,12 miliarde de operații pe secundă și pot efectua transferuri de 8,8 GB/s.
- Plăcile *Ti* 4600 pot desena 136 de milioane de triunghiuri pe secundă, pot efectua 1,23 miliarde de operații pe secundă și pot efectua transferuri de 10,4 GB/s.

GeForce 4 MX

Familia plăcilor grafice *GeForce* 4 destinate calculatoarelor ieftine cuprinde trei modele: *GeForce* 4 *MX* 420, *GeForce* 4 *MX* 440 și *GeForce* 4 *MX* 460.

- Plăcile *MX* 420 pot desena 31 de milioane de triunghiuri pe secundă, pot efectua un miliard de operații pe secundă și pot efectua transferuri de 2,7 GB/s.
- Plăcile *MX* 440 pot desena 34 de milioane de triunghiuri pe secundă, pot efectua 1,1 miliarde de operații pe secundă și pot efectua transferuri de 6,4 GB/s.
- Plăcile *MX* 460 pot desena 38 de milioane de triunghiuri pe secundă, pot efectua 1,2 miliarde de operații pe secunde și pot efectua transferuri de 8,8 GB/s.

GeForce 4 Go

Familia plăcilor grafice *GeForce* 4 destinate calculatoarelor portabile cuprinde un singur model.

Aceste plăci video pot desena 30 de milioane de triunghiuri pe secundă, pot efectua 500 de milioane de operații pe secundă și pot efectua transferuri de până la 8 GB/s.

Performanțe

Produsele care folosesc cipul *GeForce* 4 *Ti* 4600 sunt, fără îndoială, cele mai rapide plăci video disponibile în acest moment pe piață.

Acest cip ocupă locul fruntaș în familia *GeForce* 4 și, evident, este mult mai performant decât produsele din familiile anterioare.

Înaintea lansării familiei *GeForce* 4, cel mai performant cip produs de *nVidia* era *GeForce* 3 *Ti* 500. Acesta nu domina autoritar piața deoarece compania rivală *ATI* oferea cipul *Radeon* 8500 care avea performanțe comparabile.

Acum *nVidia* oferă cel mai bun produs de pe piață, cel puțin până la lansarea cipului *Radeon* 9000.

Plăcile video din familia *GeForce* 4 *MX* au performanțe mult mai scăzute decât cele din familia *GeForce* 4 *Ti*. Chiar și plăcile *GeForce* 3 *Ti* sunt mai performante. Acest lucru este explicabil deoarece *nVidia* nu ar fi dorit ca un produs ieftin (din familia *GeForce* 4 *MX*) să fie mai performant decât unul mai scump (din familia *GeForce* 3 *Ti*).

Pe piața plăcilor grafice pentru calculatoarele mobile, *nVidia* este lider incontestabil deoarece *GeForce* 4 *Go* oferă cele mai bune performanțe dintre plăcile video de acest tip.

Concluzie

Plăcile grafice *GeForce* 4 au impus un nou standard în domeniu. Probabil, în viitor acestea vor fi integrate în *chipset*-urile *nForce* pentru plăci de bază.

Borland C++BUILDER 6

Borland Software a lansat versiunea a șasea a produsului său C++Builder, completând acum cea de-a șasea generație a mediilor sale de programare. La fel ca Borland JBuilder 6 și Borland Delphi 6, noul Borland C++Builder 6 oferă numeroase facilități pentru programatori, motiv pentru care a devenit produsul acestei luni.

Lansare

După lansările din 2001 ale produselor *JBuilder* 6 și *Delphi* 6, singurul mediu de programare oferit de *Borland* care nu ajunsese la versiunea 6 era *C++Builder*. Acest "neajuns" a fost "remediat" pe data de 5 februarie 2002, când compania a anunțat disponibilitatea mediului de programare *Borland C++Builder* 6.

Prezentare

Borland C++Builder 6 este disponibil în trei variante, și anume *C++Builder* 6 *Personal*, *C++Builder* 6 *Professional* și *C++Builder* 6 *Enterprise*.

Produsul oferă un mediu de programare vizual care permite, pe lângă dezvoltarea aplicațiilor *Windows* clasice, crearea de servicii *Web* sau de aplicații capabile să ruleze pe diferite sisteme de operare. De asemenea, pot fi create aplicații pentru platforma *Microsoft .NET*.

C++Builder 6 oferă suport pentru ultimele tehnologii apărute cum ar fi *XML*, *SOAP* sau *WSDL*.

În viitor se preconizează lansarea unei versiuni *C++Builder* pentru sistemul de operare *Linux*; în acest moment există o variantă *Delphi* pentru *Linux* care poartă denumirea *Kylix* și a ajuns la versiunea a doua.

Sistemul necesar

Pentru a putea fi folosit, *C++Builder* 6 necesită 128 MB RAM (este recomandabil să aveți la dispoziție cel puțin 256 MB), o unitate *CD-ROM*, o placă video care să suporte o rezoluție de 800 x 600 cu cel puțin 256 de culori și un *mouse* sau orice alt dispozitiv similar. Pentru a realiza o instalare completă a versiunii *Enterprise* este necesar un spațiu liber pe disc de 750 MB. Versiunea *Professional* necesită 650 MB, în timp ce versiunea *Personal* necesită 550 MB.

Sistemele de operare pe care poate rula acest produs sunt Microsoft *Windows* 98, *Microsoft Windows Me*, *Microsoft Windows* 2000 (cu *Service Pack* 2) și *Microsoft Windows XP*. Nu este oferit suport pentru sistemele de operare *Microsoft Windows* 95 și *Microsoft Windows NT* 4.0.

Prețuri

Pachetul *Borland C++Builder* 6 *Enterprise* poate fi achiziționat la prețul de 2999$. Prețul pachetului *Borland C++Builder* 6 *Professional* este de 999$, în timp ce prețul pachetului *Borland C++Builder* 6 *Personal* este de 69$.

Aceste prețuri sunt valabile pentru clienții noi. Cei care au achiziționat versiunea *Borland C++Builder* 5 beneficiază de anumite reduceri. De asemenea, sunt oferite reduceri pentru instituțiile educaționale care folosesc *C++Builder*.

O versiune de încercare a pachetului *Borland C++Builder* 6 *Enterprise* este disponibilă pentru *download* la adresa **www.borland.com/downloads**. Dacă descărcați acest produs, timp de 60 de zile, veți putea beneficia de facilitățile oferite de acest mediu de programare.

Borland vs. Microsoft

Așadar, compania *Borland Software* a completat la începutul anului 2002 cea de-a șasea generație a mediilor sale de programare.

Compania rivală *Microsoft* oferea de peste trei ani pachetul de programe *Visual Studio* 6 care cuprinde mai multe medii de programare cum ar fi *Visual Basic*, *Visual C++*, *Visual J++* etc.

Este foarte probabil ca avantajul psihologic al celor de la *Microsoft* să fie păstrat deoarece cea de-a șaptea versiune a pachetului *Visual Studio* este lansată în prima jumătate a lunii februarie.

Mediile de programare *Microsoft Visual C++* 7.0, *Microsoft Visual Basic* 7.0 etc. sunt componente ale pachetului de programe *Microsoft Visual Studio .NET*.

Concluzii

Chiar dacă principalul său competitor în domeniul mediilor de programare este gigantul *Microsoft*, compania *Borland* oferă produse de foarte bună calitate care sunt folosite de milioane de programatori din întreaga lume. Principalul avantaj al produselor *Borland* este facilitatea de a crea aplicații independente de platformă.

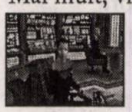
STAR TREK

~ Bridge Commander ~

La începutul lunii februarie a fost lansat un nou joc inspirat din universul Star Trek. Aceasta este primul simulator spațial a cărui acțiune este inspirată de seria Star Trek, the Next Generation. Bridge Commander este un produs foarte spectaculos, motiv pentru care a primit titlul de joc al lunii februarie.

Firul epic

În timpul unei misiuni care se desfășura în apropierea unei stele, astrul explodează și unda de șoc care se formează afectează destul de grav nava în care vă aflați, ucigându-l pe căpitan. Mai mult, viața coloniștilor de pe una



dintre planetele care se învârt în jurul stelei este pusă în pericol.

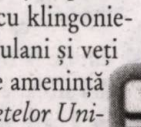
Va trebui să preluați controlul navei stelare și să vă conduceți echipajul în încercarea de a rezolva misterul. Sarcina voastră este foarte simplă: va trebui să descoperiți motivele dezastrului, precum și o modalitate de a evita alte explozii de acest gen.

De-a lungul jocului veți lupta cu cardassienii, vă veți alia cu klingonienii, îi veți spiona pe romulani și veți descoperi un pericol care amenință existența *Federației Planetelor Unite*.

Jocul constă în 30 de misiuni diferite care includ elementele fundamentale întâlnite în universul *Star Trek*: apărare, luptă, diplomație, explorare, cercetări științifice, misiuni de salvare etc.

În timpul jocului veți putea ajunge la comanda navelor stelare din clasele *Galaxy* și *Sovereign*.

Veți fi capabili să vă salvați echipajul, să salvați coloniștii și să apărați *Federația* în fața noii amenințări? Răspunsul îl veți găsi doar jucând *Star Trek - Bridge Commander*.



Despre joc

Star Trek - Bridge Commander este un simulator spațial creat de compania *Totally Games* și distribuit de *Activision*.

Designer-ul principal al jocului este *Larry Holland*, cel care a proiectat celebrele simulatoare spațiale din seria *Star Wars - X-Wing* produse tot de compania *Totally Games*.

El spune despre noul său proiect că nu se aseamănă cu alte jocuri inspirate din universul *Star Trek*. În *Bridge Commander* jucătorul se va afla la comanda celei mai puternice nave stelare din *Galaxie* și o va controla cu ajutorul echipajului. Interfața este foarte simplă, fiind bazată pe principiul *point and click*. Jucătorii pasionați pot profita de opțiunea de a prelua consola tactică în timpul misiunilor.

Producătorii s-au concentrat foarte mult asupra bătăliilor, strategia aleasă având un impact decisiv asupra desfășurării acestora.

Jocul se bazează pe un nou motor de redare grafică numit *Netlmmerse* care permite obținerea unor efecte deosebit de interesante și extrem de spectaculoase. *Engine*-ul jocului a fost construit pe baza acestui motor de redare grafică, ceea ce a permis obținerea unor performanțe grafice excelente.

Universul Star Trek pe PC

Serialul de televiziune *Star Trek* a inspirat multe jocuri pentru *PC*-uri. Din nefericire, majoritatea acestora nu s-au ridicat la nivelul așteptărilor.

În ultimii ani s-a observat o îmbunătățire substanțială a calității jocurilor inspirate din universul *Star Trek*. Titluri cum ar fi *Starfleet Command*, *Star Trek Armada* sau *Star Trek Voyager: Elite Force* au atras mulți jucători pasionați, chiar dacă aceștia nu erau atrași și de seria televizivă. Klingonienii, romulanii, ferengii sau borgii au intrat în viața unora doar prin intermediul *PC*-ului.

Star Trek - Bridge Commander este un nou titlu de succes care va fi jucat cu plăcere de către toți cei pasionați de universul *Star Trek*, dar și de cei care doresc să se bucure de un simulator spațial de calitate.

Cine știe, CÂȘTIGĂ

Numeroși utilizatori ai Internetului caută diferite site-uri care să le permită câștigarea unor premii. Primul site românesc care vă oferă posibilitatea de a vă pune în valoare cunoștințele de cultură generală pentru a câștiga anumite premii este www.cinestiecastiga.ro; acesta este site-ul pe care vi-l vom prezenta în această lună.

Prezentare

La **www.cinestiecastiga.ro** veți găsi un *site* dedicat testării cunoștințelor dumneavoastră de cultură generală.

După ce vă veți înscrie, veți avea dreptul să jucați. Inițial veți avea zece puncte de joc (veți putea juca de zece ori) și în fiecare zi veți primi câte cinci puncte. În cazul în care, într-o anumită zi, nu veți utiliza toate punctele, acestea vor fi reportate în ziua următoare. Totuși, există o limită pentru punctele care pot fi reportate: nu veți putea avea niciodată mai mult de zece puncte.

Un joc constă din cinci niveluri; la fiecare dintre acestea va trebui să răspundeți la cinci întrebări cu un grad de dificultate corespunzător nivelului. Răspunzând corect la întrebări, puteți acumula unități *C$C* pe care, ulterior, le puteți transforma în premii.

Pentru întrebările de la primul nivel nu veți obține unități *C$C*. Un răspuns corect la al doilea nivel vă aduce o unitate *C$C*, pentru un răspuns corect la al treilea nivel sunteți premiați cu zece unități *C$C*. La nivelurile următoare veți primi o sută, respectiv, o mie de unități *C$C* pentru fiecare răspuns corect.

Un răspuns incorect va duce la pierderea tuturor punctelor câștigate la nivelul respectiv, dar nu și a celor câștigate la nivelurile anterioare.

Pentru fiecare întrebare veți avea la dispoziție doar 30 de secunde pentru a răspunde, expirarea timpului ducând la pierderea punctelor obținute la nivelul curent și la terminarea jocului.

Pentru fiecare întrebare veți avea patru variante de răspuns, dintre care va trebui să o alegeți pe cea corectă.

Pe parcursul unui joc veți putea folosi trei elemente ajutătoare: `/2` (eliminarea a două dintre variante), `×2` (dublarea timpului) și `-2` (schimbarea întrebării curente, dar cu pierderea a două puncte de joc).

În orice moment vă puteți retrage; nu veți pierde unitățile *C$C* câștigate până în acel moment, dar jocul se va încheia.

Premii

După acumularea unui anumit număr de unități *C$C* veți putea să vă alegeți unele premii. Cele disponibile în acest moment sunt:

- casete audio cu muzică românească (200 de unități *C$C*);
- casete audio cu muzică internațională (400 de unități *C$C*);
- carduri pentru acces *Internet* (400 de unități *C$C*);
- *CD*-uri cu muzică românească sau internațională (1000 de unități *C$C*);
- cărți (1000 de unități *C$C*);
- jocuri *PC* (2000 de unități *C$C*);
- casete video (3000 de unități *C$C*).

Întrebări

Site-ul vă permite să vă testați cunoștințele în cele mai diverse domenii de la literatură, geografie, istorie sau mitologie până la fizică, chimie, biologie etc.

Din nefericire, unele răspunsuri considerate corecte sunt eronate. De exemplu, creatorii *site*-ului consideră că a fost de mult modificată *Constituția României* și țara noastră a devenit republică parlamentară. De asemenea, veți putea afla că *Mihai Eminescu* s-a născut fie la *Botoșani*, fie la *Ipotești*. O altă situație interesantă este aceea în care veți fi întrebați în ce județ se află un anumit oraș și veți putea alege între opțiunile *Iași*, *Neamț*, *Iași* și *Bacău*. Evident, răspunsul corect este *Iași*, dar va trebui să fiți foarte inspirați în momentul în care veți alege unul dintre cele două răspunsuri care, după părerea dumneavoastră, sunt ambele corecte.

Securitate

Din nefericire, *site*-ul are unele probleme de securitate; puteți intra în alt cont chiar și "din greșeală".

Bursele AGORA

A doua etapă a concursului de programare Bursele Agora s-a încheiat. În cadrul acestui număr vă vom prezenta soluțiile problemelor de la această rundă, clasamentul etapei, clasamentul general după două etape, precum și cele trei probleme ale rundei a patra.

A doua rundă

Problemele celei de-a doua runde au fost mai puțin dificile decât cele propuse pentru rezolvare în cadrul primei etape. Chiar dacă pentru problema *Funcție* nici un concurent nu a obținut punctajul maxim, celelalte două probleme au fost rezolvate corect de foarte mulți concurenți.

Vă prezentăm în continuare punctajele primilor 20 clasați în clasamentul acestei runde:

1. Adrian-Nicolae Cârcu, Bistrița	*293*
2. Mugurel-Ionuț Andreica, București	*292*
3. Adrian Balașko, Zalău	*288*
4. Tiberiu Dăneț, București	*272*
4. Silvestru-Cosmin Negruțeri, Bistrița	*272*
6. Kovychev Ruslan Alexanderovich, Rusia	*268*
7. Paul Rusu, Cluj-Napoca	*241*
8. Radu-Andrei Ștefan, Brașov	*238*
9. Andrei Homescu, Târgu-Jiu	*234*
10. Csaba András, Oradea	*232*
11. Radu Berinde, București	*230*
11. Emilian Miron, Bacău	*230*
13. Octavian-Daniel Dumitran, București	*229*
14. Andrei Vancea, Cluj-Napoca	*227*
15. Maximilian Machedon, București	*226*
16. Bogdan-Vasile Hârjoc, Cugir	*213*
17. Bogdan Nicolae, Sibiu	*208*
18. Cornel Mihăilă, Brașov	*201*
19. Victor-Marius Costan, București	*200*
19. Csaba Pătcaș, Oradea	*200*

Runda a cincea

Cele trei probleme ale rundei a cincea sunt *Fibonacci*, *Atlantis* și *Tic-Tac-Toe*.

Prima dintre ele a fost propusă de *Claudiu Soroiu*, student în anul II la *Universitatea Babeș-Bolyai* din Cluj și redactor al revistei noastre.

A doua problemă a fost propusă de *Mihai Stroe*, student în anul IV la *Universitatea Politehnica* din București, fost olimpic internațional, câștigător al primei ediții a concursului nostru și clasat pe locul al doilea la ediția de anul trecut.

Ultima problemă a fost propusă de *Mihai Oltean*, doctorand și master al *Universității Babeș-Bolyai* din Cluj, autor a mai multor cărți de specialitate printre care și câteva culegeri de probleme.

Rezolvările celor trei probleme pot fi trimise până la data de *31 martie 2002*. Rezultatele oficiale ale rundei a cincea vor deveni publice pe data de *15 aprilie 2002*.

Pentru a evita întârzierile, vă recomandăm să trimiteți soluțiile cu câteva ore sau zile înaintea expirării termenului limită.

La mulți ani!

Dorin-Daniel Dudău, Târgu-Jiu - 1 II 1984
Andrei Markovits, Satu Mare - 2 II 1983
Theodor Drăgoi, Pitești - 2 II 1985
Ion Ionașcu, București - 4 II 1983
Mihai-Silviu Chivu, Brăila - 7 II 1984
Dorin-Cătălin Stoicescu, Buzău - 7 II 1984
Grigore-Mihai Dobra, Baia Mare - 10 II 1985
Mihai Pantelimon, București - 13 II 1984
Ștefan Conțiu, Reghin - 15 II 1984
Ionuț Stan, Aiud - 15 II 1987
Andrei-Daniel Turi, Dr. Turnu Severin - 16 II 1984
Adrian-Nicolae Cârcu, Bistrița - 17 II 1981
Radu Vatavu, Suceava - 17 II 1981
Cătălin Bujdei, Brașov - 17 II 1982
Cornel Mihăilă, Brașov - 17 II 1982
Andrei Homescu, Târgu-Jiu - 19 II 1985
Andrei Blaj, Suceava - 20 II 1985
Ștefan Panțiru, Bicaz-Chei - 22 II 1984
Cătălin Bogda, Buzău - 24 II 1985
Eduard Dumitru, Pitești - 25 II 1985
Ramona Cazacu, Vaslui - 26 II 1985
Marius Anton, Alba Iulia - 29 II 1984

Runda 2, problema 1

MATRICE

La această problemă, 29 dintre participanți au obținut punctajul maxim, deci problema nu a fost foarte dificilă. Metoda folosită de către autor pentru rezolvarea problemei este *programarea dinamică*.

Evident, determinarea fiecărui element din matrice consumă prea mult timp, deci nu este o soluție viabilă. Rezolvarea propusă se bazează pe o tehnică recursivă.

Vom încerca să determinam, pentru fiecare linie din M_7, numărul de elemente cu valoarea 1 aflate pe linia respectivă. Folosind această informație, linia pe care se află al X-lea element poate fi se găsită imediat (se parcurg liniile matricei M_7, se însumează valorile corespunzătoare primelor K linii și, în cazul în care această sumă este mai mică decât X se trece la linia K+1; dacă suma este mai mare sau egală cu X, atunci al X-lea element se află pe linia K).

Se pune problema determinării numărului de elemente cu valoarea 1 de pe o linie a matricei M_7. Linia respectivă este obținută prin expandarea unei linii a matricei M_6 (de fapt prin această expandare se obțin N linii, linia respectivă fiind una dintre ele). Dacă se cunoaște numărul de elemente cu valoarea 1 de pe linia corespunzătoare a matricei M_6, rezultatul cerut se determină ușor.

Pentru simplificarea formulelor vom numerota liniile și coloanele începând cu 0; în final se vor incrementa cu 1 valorile obținute în urma calculelor.

Așadar, dorim să determinăm numărul elementelor cu valoarea 1 de pe o linie L a matricei M_7. Pentru aceasta, trebuie să cunoaștem numărul elementelor cu valoarea 1 de pe linia $[L / N]$ a matricei M_6. Fiecare dintre aceste elemente se transformă în matricea A și "contribuie" la construirea liniei L din M_7 prin intermediul liniei *rest*$[L / N]$ a matricei A. Similar, fiecare element cu valoarea 0 din M_6 se transformă în matricea B. Așadar, rezultatul cerut este:

numărul de elemente cu valoarea 1 *de pe linia* $[L / N]$ *a matricei* M_6 * *numărul de elemente cu valoarea* 1 *de pe linia* rest$[L / N]$ *din matricea* A + *numărul de elemente cu valoarea* 0 *de pe linia* $[L / N]$ *a matricei* M_6 * *numărul de elemente cu valoarea* 1 *de pe linia* rest$[L / N]$ *din matricea* B.

Pentru a determina numărul de elemente cu valoarea 1 de pe linia $[L / N]$ a matricei M_6 vom aplica recursiv același procedeu, până ajungem la $M_1 = A$. Din nefericire, această metodă duce la repetarea unor calcule de foarte multe ori, prin apelarea funcției recursive cu aceiași parametri.

Pentru a rezolva acest impediment se memorează rezultatele într-o matrice NR de dimensiuni $6 \times n^6$ în care elementul NR_{ij} reprezintă numărul de elemente cu valoarea 1 de pe linia j a matricei M_i.

Au rezolvat corect:

András Csaba, Oradea
Mugurel-Ionuț Andreica, București
Adrian Balașko, Zalău
Radu Berinde, București
Vencel-Istvan Bors, Oradea
Adrian-Nicolae Cârcu, Bistrița
Mihai-Daniel Cilidariu, Botoșani
Victor-Marius Costan, București
Tiberiu Dănet, București
Radu Dondera, București
Octavian-Daniel Dumitran, București
Ștefan Gheorghe, București
Ioan-Claudiu Gruia, București
Andrei Homescu, Târgu-Jiu
Liviu Lalescu, Craiova
Maximilian Machedon, București
Cornel Mihăilă, Brașov
Emilian Miron, Bacău
Silvestru-Cosmin Negruțeri, Bistrița
Mihai Pantelimon, București
Csaba Pătcaș, Oradea
Mihail Popa, București
Claudiu Raicu, Pitești
Constantin Răuțu, Botoșani
Kovychev Ruslan Alexanderovich, Rusia
Paul Rusu, Cluj-Napoca
Radu Andrei Ștefan, Brașov
Victoria Tarasova, Ucraina
Andrei Vancea, Cluj-Napoca

Rezultatele pentru M_7 nu sunt memorate, deoarece fiecare dintre acestea sunt calculate o singură dată. Fără această optimizare, matricea NR nu ar fi putut fi memorată în memoria convențională.

După determinarea liniei pe care se află al X-lea element cu valoarea 1, determinarea coloanei este simplă și poate fi realizată fie într-o manieră similară, fie pur și simplu determinând pe rând fiecare element de pe linia respectivă a matricei M_7. Pe linia respectivă se află cel mult $5^7 = 78125$ elemente, deci numărul iterațiilor nu ar fi foarte mare.

Codul sursă al programului este disponibil pentru *download* la adresa **www.ginfo.ro/concurs/matrice2.shtml**.

Runda 2, problema 2

VÂNZĂRI

Această problemă a fost rezolvată corect de 18 dintre participanții la a doua rundă a concursului *Bursele Agora*, ceea ce demonstrează faptul că nu este greu de rezolvat.

Ea este doar o particularizare a problemei ***Telefoane mobile*** propusă spre rezolvare celor care au participat la ediția din anul 2001 a *Olimpiadei Internaționale de Informatică*.

În numărul 11/7 (noiembrie 2001) a apărut prezentarea soluției oficiale a problemei de la *IOI*, atât pentru cazul unidimensional (care corespunde problemei ***Vânzări***), cât și în cazul bidimensional (care corespunde problemei ***Telefoane mobile***).

Prima idee de rezolvare a problemei ***Vânzări*** este de a păstra un șir a ale cărui elemente a_i conțin valoarea vânzărilor către magazinele cuprinse între pozițiile 1 și i.

Astfel, suma vânzărilor către magazinele cuprinse între pozițiile i și j poate fi obținută printr-o simplă diferență de tipul $a_j - a_{i-1}$. Pentru a nu apărea dificultăți în cazul în care trebuie determinată suma vânzărilor către magazinele cuprinse între 1 și i, vom inițializa elementul a_0 cu valoarea 0. Vom avea $a_i - a_{1-1} = a_i - a_0 = a_i - 0 = a_i$.

În cazul în care are loc o vânzare către un magazin i, atunci trebuie actualizate primele i elemente ale vectorului a. Așadar, în cazul cel mai defavorabil, ordinul de complexitate al unei actualizări este $O(N)$. Pentru aflarea valorii totale a vânzărilor către magazinele cuprinse între pozițiile i și j, se efectuează o simplă scădere, deci ordinul de complexitate este $O(1)$. Vom avea maximum K actualizări, deci ordinul de complexitate al algoritmului este $O(N \cdot K)$.

Se observă că un algoritm cu acest ordin de complexitate nu se încadrează în limita de timp admisă pentru cazul cel mai defavorabil. Pentru a găsi un algoritm eficient trebuie găsită o modalitate de păstrare a datelor care să permită realizarea mai rapidă a actualizărilor.

Structura de date propusă poartă denumirea de ***arbore indexat binar***. Aceasta a fost prezentată în numărul 11/7 al *GInfo*. Articolul este disponibil pentru *download* la adresa **www.ginfo.ro/revista/11_7/index.shtml**.

Folosind această structură, actualizarea datelor este mult mai rapidă, ordinul de complexitate al algoritmului fiind, în cazul cel mai defavorabil, $O(\log N)$.

Așa cum s-a arătat în articolul amintit anterior, suma corespunzătoare magazinelor cuprinse între pozițiile i și j nu mai poate fi determinată în timp constant, ordinul de complexitate al acestei operații fiind, în cazul cel mai defavorabil, $O(\log N)$.

Au rezolvat corect:

Csaba András, Oradea
Mugurel-Ionuț Andreica, București
Adrian Balașko, Zalău
Andrei Benea, Focșani
Radu Berinde, București
Adrian-Nicolae Cârcu, Bistrița
Victor-Marius Costan, București
Tiberiu Dănet, București
Octavian-Daniel Dumitran, București
Ștefăniță Fechete, Suceava
Ștefan Gedo, București
Bogdan-Vasile Hârjoc, Cugir
Andrei Homescu, Târgu-Jiu
Liviu-Laurențiu Iacob, București
Emilian Miron, Bacău
Silvestru-Cosmin Negruțeri, Bistrița
Bogdan Nicolae, Sibiu
Kovychev Ruslan Alexanderovich, Rusia

Se observă că primul algoritm prezentat este mai lent la actualizarea datelor, dar este mai rapid la regăsirea acestora. Din aceste motive, în cazul în care se efectuează puține vânzări și se cer multe statistici, primul algoritm ar fi mai rapid. În cazul cel mai favorabil (când nu se efectuează nici o actualizare) ordinul de complexitate al algoritmului este $O(N)$.

Totuși, dacă se folosește al doilea algoritm, operațiile de actualizare și regăsire a datelor necesită doar un timp logaritmic. Deoarece au loc K astfel de operații, ordinul de complexitate este $O(K \cdot \log N)$. Acesta nu depinde de tipul operațiilor (vânzări sau cereri de statistici) pentru că timpul necesar efectuării acestora are același ordin de complexitate. Așadar, ordinul de complexitate al algoritmului nu variază, indiferent de structura datelor de intrare.

În medie, numărul vânzărilor este aproximativ egal cu cel al statisticilor cerute, deci vor fi aproximativ $K / 2$ actualizări. Ordinul de complexitate al primului algoritm va fi, în cazul mediu, $O(K \cdot N)$. Ca urmare, și pentru cazul mediu, al doilea algoritm este mai performant.

Putem trage concluzia că este mai indicată folosirea structurii de *arbore binar indexat* chiar dacă, în anumite situații, folosirea vectorului de sume ar putea duce la obținerea unor performanțe superioare.

Codul sursă al programului este disponibil pentru *download* la adresa **www.ginfo.ro/concurs/vanzari.shtml**.

Runda 2, problema 3

FUNCȚIE

Se știe că rezolvarea ecuațiilor este o problemă dificilă a matematicii. O dovadă în plus în acest sens este faptul că nici unul dintre participanți nu a reușit să obțină toate cele 100 de puncte corepunzătoare acestei probleme.

Cele mai mari punctaje au fost obținute de ***Adrian-Nicolae Cârcu*** din *Bistrița* (93 de puncte), ***Mugurel-Ionuț Andreica*** din *București* (92 de puncte) și ***Adrian Balașko*** din *Zalău* (88 de puncte).

Nu se știe dacă există metode sigure pentru determinarea soluțiilor unei ecuații în cazul general. Pentru anumite cazuri particulare există algoritmi siguri care determină exact toate soluțiile. De exemplu, există metode sigure pentru determinarea soluției ecuațiilor polinomiale de gradul unu, doi, trei sau patru. Însă, pentru ecuații polinomiale de grad mai mare s-a demonstrat că nu există nici o metodă exactă pentru determinarea tuturor soluțiilor. De asemenea, precizia cu care este determinată o soluție este o problemă dificil de rezolvat.

Din aceste motive, pentru rezolvarea acestei probleme trebuie folosit un algoritm euristic. Acesta nu va garanta întotdeauna obținerea tuturor soluțiilor, dar folosirea lui în condiții optime ar duce la obținerea unui număr mare de soluții, cu o precizie bună.

Unul dintre cei mai simpli algoritmi aproximativi folosiți pentru rezolvarea unei ecuații este ***metoda înjumătățirii intervalului***. Acest algoritm are dezavantajul că pe intervalul pe care este aplicat trebuie ca ecuația să aibă exact o soluție. În caz contrar, aplicarea lui poate duce la obținerea de rezultate imprevizibile. Dacă există mai multe rădăcini ale ecuației în intervalul dat, atunci ar trebui să împărțim intervalul în subintervale în interiorul cărora există o singură rădăcină, iar apoi să aplicăm metoda înjumătățirii intervalului pentru fiecare dintre aceste subintervale. Această metodă presupune însă împărțirea unui interval în subintervale care conțin exact o soluție a ecuației, ceea ce, din punct de vedere al gradului de dificultate, echivalează cu rezolvarea problemei.

O altă idee de rezolvare este parcurgerea întregului interval cu pasul 10^{-5} și afișarea valorilor care ar fi putut constitui soluția problemei. Această abordare este eficientă doar în cazul în care lungimea intervalului este mică, dar ineficientă pentru intervale de lungime mare.

O altă idee este aplicarea unei ***strategii evolutive***. Se pornește cu o populatie inițială care va evolua doar cu ajutorul operatorului de *mutație*. Fiecare individ va fi selectat pentru mutație, iar fiul rezultat în urma aplicării operatorului va fi comparat direct cu părintele său. Cel mai performant dintre cei doi indivizi (părinte și fiu) va fi inclus în noua populație. Calitatea (*fitness*-ul) unui individ x se stabilește în funcție de distanța la care se află $f(x)$ față de axa Ox. Cu alte cuvinte, calitatea unui individ x este chiar $|f(x)|$. Prin mutație, se înțelege adăugarea sau scăderea unei valori mici la (de la) valoarea individului. În final sunt afișate soluțiile aflate la o distanță mai mare decât precizia dată unele de altele. Această soluție are dezavantajul unei execuții lente cauzată de faptul că unii indivizi ajung mai repede decât alții la soluție și, cu toate acestea, ei vor fi supuși din nou mutației. Eliminarea celor care au ajuns deja la soluții acceptabile ar însemna efectuarea unor operații suplimentare, deci o creștere a complexității algoritmului.

Din aceste motive, se va aplica o variantă modificată a algoritmului de mai sus. Se pornește cu o populație inițială generată aleator. Fiecare individ este selectat și asupra lui se aplică operatorul de mutație până în momentul în care individul ajunge să fie soluție a ecuației. Pasul de mutație este 10^{-5}. Această valoare este exact precizia cu care trebuie determinate soluțiile. Din nefericire, această valoare este destul de mică pentru unele cazuri.

Din acest motiv vom modifica din nou algoritmul. Pasul de mutație nu va mai fi fix, ci va descrește treptat. La început, pasul de mutație va fi mai mare (de exemplu 10^{-3}). Acest pas de mutație va face ca soluțiile să efectueze salturi mai mari spre soluție. În momentul în care nu se vor mai putea efectua mutații cu acest pas, acesta va fi micșorat (noua valoare ar putea fi 10^{-4}) și algoritmul va continua. Acest tip de strategie de căutare se numește ***călire simulată***. Pasul de mutație este scăzut pe măsura ce crește numărul generațiilor.

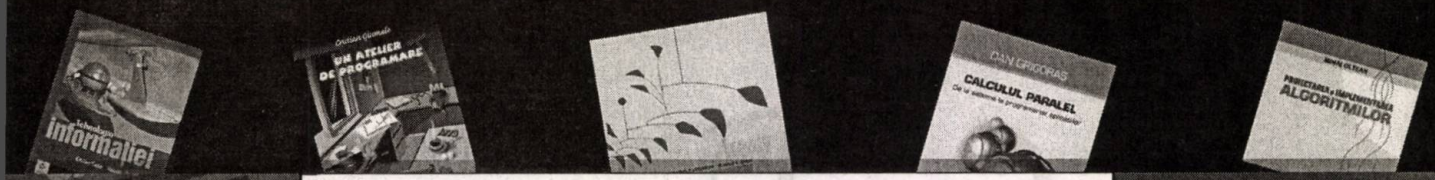
Nici acest algoritm nu este sigur. Datorită faptului că pasul de mutație este, inițial, mai mare decât precizia cu care trebuie determinate soluțiile, s-ar putea ca unele soluții să fie pierdute. O îmbunătățire ar putea fi setarea pasului inițial al mutației în funcție de lungimea intervalului pe care se caută soluțiile ecuației. Astfel, pentru lungimi mari ale intervalului se va începe cu un pas mare de mutație, iar pentru lungimi mici ale intervalului se va începe cu pași mici de mutație. Pentru mărirea șanselor de a găsi toate soluțiile vom genera o populație inițială cât mai mare.

Schema codului sursă al programului este disponibilă pentru *download* la **www.ginfo.ro/concurs/functie.shtml**.

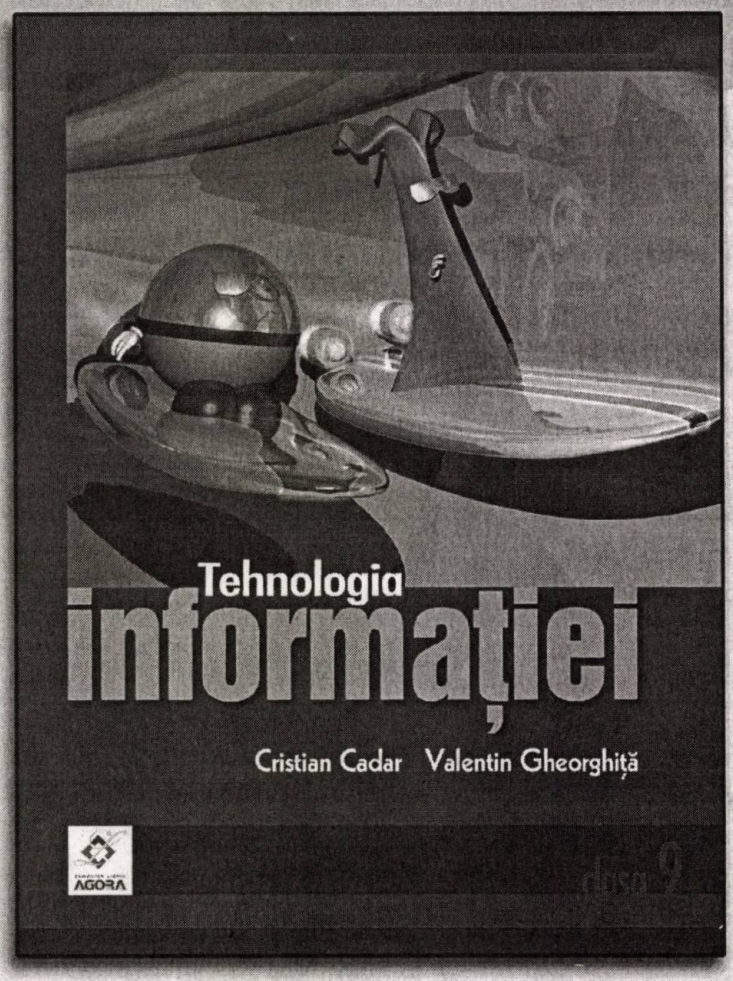
GInfo Top 50

1.	*Mugurel-Ionuț Andreica, București*	543
2.	*Adrian-Nicolae Cârcu, Bistrița*	514
3.	*Cosmin-Silvestru Negrușeri, Bistrița*	490
4.	*Tiberiu Dăneț, București*	455
5.	*Adrian Balașko, Zalău*	445
5.	*Csaba András, Oradea*	445
7.	*Radu Berinde, București*	411
8.	*Bogdan Nicolae, Sibiu*	400
9.	*Andrei Vancea, Cluj-Napoca*	391
10.	*Ștefăniță Fechete, Suceava*	383

11.	*Octavian-Daniel Dumitran, București*	377	
12.	*Andrei Homescu, Târgu-Jiu*	371	
13.	*Radu-Andrei Ștefan, Brașov*	368	
14.	*Ioan-Claudiu Gruia, București*	363	
15.	*Maximilian Machedon, București*	361	
16.	*Csaba Pătcaș, Oradea*	355	
17.	*Claudiu Raicu, Pitești*	337	
18.	*Emilian Miron, Bacău*	328	
19.	*Paul Rusu, Cluj-Napoca*	307	
20.	*Radu Dondera, București*	286	
20.	*Liviu-Laurențiu Iacob, București*	286	
22.	*Andrei Benea, Focșani*	280	
23.	*Ștefan Gheorghe, București*	268	
23.	*Kovychev Ruslan Alexanderovich, Rusia*	268	
25.	*Constantin Răuțu, Botoșani*	262	
26.	*Liviu Lalescu, Craiova*	245	
27.	*Alpár-Ferenc Perini, Sfântu Gheorghe*	241	
28.	*Mihai Pantelimon, București*	240	
29.	*Liviu-Cosmin Andreicuț, Bacău*	235	
30.	*Vadim Barkalov, S.U.A.*	231	
31.	*Mihai-Daniel Cilidariu, Botoșani*	229	
32.	*Bogdan-Vasile Hârjoc, Cugir*	216	
33.	*Ștefan Bucur, București*	212	
34.	*Mihail Popa, București*	204	
35.	*Cornel Mihăilă, Brașov*	201	
35.	*Sorin Otescu, Brașov*	201	
37.	*Andrei David, Râmnicu Vâlcea*	200	
37.	*Victor-Marius Costan, București*	200	
39.	*Aurelian Marinescu, Brașov*	192	
39.	*Cristian Bojinovici, București*	192	
41.	*Slotta Johannes, Germania*	190	
42.	*Victoria Tarasova, Ucraina*	180	
43.	*Vlad Dascălu, Bacău*	174	
43.	*Dan Ghinea, București*	174	
45.	*Cristian Toth, Lugoj*	164	
46.	*Alexandru Palade, Târgu-Mureș*	159	
46.	*Dorin-Daniel Dudău, Târgu-Jiu*	159	
48.	*Silviu-Gabriel Udrea, Târgu-Jiu*	157	
49.	*Cosmin Lehene, Zalău*	156	
50.	*Dan-Octavian Savu, București*	155	



COMPUTER LIBRIS AGORA



Lucrarea de față își propune să pregătească cititorul în scopul utilizării unui calculator personal. Ea urmărește programa școlară, aprobată de Ministerul Educației Naționale, pentru disciplina "Tehnologia informației", disciplină în planul de învățământ a claselor a IX-a la profilul teoretic și pedagogic. Manualul corespunde și programei de la disciplina "Informatică - Tehnologii asistate pe calculator", profilul tehnologic.

Pentru parcurgerea acestei cărți nu sunt necesare cunoștințe prealabile în domeniu, termenii și noțiunile prezentate sunt explicate detaliat în carte. Exemplele sunt sugestive, cartea nu este încărcată inutil cu informații care depășesc programa amintită.

COMPUTER LIBRIS AGORA

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

FIBONACCI

Denumirea subdirectorului: **FIBO**

Denumirea fișierului de intrare: **FIBO.IN**

Denumirea fișierului de ieșire: **FIBO.OUT**

Descrierea problemei

Se consideră celebrul șir al lui *Fibonacci*, definit astfel:

- $F_0 = 0$;
- $F_1 = 1$;
- $F_i = F_{i-1} + F_{i-2}$ pentru $i > 1$.

Sarcina voastră este de a determina indicele unui element al acestui șir care este divizibil cu un număr de forma 2^n.

Valoarea indicelului trebuie să fie mai mică decât 2^n. De exemplu, pentru $n = 3$, unul dintre elementele divizibile cu 2^3 este 8 (are indicele 6 și $6 < 8$).

Date de intrare

Fișierul de intrare conține o singură linie pe care se află valoarea n.

Date de ieșire

Fișierul de ieșire va conține o singură linie pe care se va afla indicele elementului divizibil cu 2^n.

În cazul în care nu există nici un element divizibil cu 2^n a cărui indice să aibă o valoare mai mică decât 2^n, atunci în fișierul de ieșire se va scrie valoarea -1.

Restricție

- $1 \le n \le 10000$.

Exemple

FIBO.IN	FIBO.OUT
3	6

FIBO.IN	FIBO.OUT
1	-1

FIBO.IN	FIBO.OUT
5	24

FIBO.IN	FIBO.OUT
2	-1

Explicație

Primii termeni ai șirului lui *Fibonacci* sunt:

- $F_0 = 0$;
- $F_1 = 1$;
- $F_2 = 0 + 1 = 1$;
- $F_3 = 1 + 1 = 2$;
- $F_4 = 1 + 2 = 3$;
- $F_5 = 2 + 3 = 5$;
- $F_6 = 3 + 5 = 8$;
- $F_7 = 5 + 8 = 13$;
- $F_8 = 8 + 13 = 21$;
- $F_9 = 13 + 21 = 34$;
- $F_{10} = 21 + 34 = 55$;
- $F_{11} = 34 + 55 = 89$;
- $F_{12} = 55 + 89 = 144$;
- $F_{13} = 89 + 144 = 233$;
- $F_{14} = 144 + 233 = 377$;
- $F_{15} = 233 + 377 = 610$;
- $F_{16} = 377 + 610 = 987$;
- $F_{17} = 610 + 987 = 1597$;
- $F_{18} = 987 + 1597 = 2584$;
- $F_{19} = 1597 + 2584 = 4181$;
- $F_{20} = 2584 + 4181 = 6765$;
- $F_{21} = 4181 + 6765 = 10946$;
- $F_{22} = 6765 + 10946 = 17711$;
- $F_{23} = 10946 + 17711 = 28657$;
- $F_{24} = 17711 + 28657 = 46368$.

Un element divizibil cu $2^3 = 8$ este 8; indicele acestuia are valoarea 6; această valoare este mai mică decât 8. Cel mai mic element divizibil cu $2^1 = 2$ este 2; indicele acestuia are valoarea 3; această valoare este mai mare decât 2. Un element divizibil cu $2^5 = 32$ este 46368; indicele acestuia are valoarea 24; această valoare este mai mică decât 32. Cel mai mic element divizibil cu $2^2 = 4$ este 8. Indicele acestuia are valoarea 6; această valoare este mai mare decât 4.

Timp de execuție/test: **2 secunde**

Runda 5, problema 2

ATLANTIS

Denumirea subdirectorului: **ATLANTIS**
Denumirea fișierului de intrare: **ATLANTIS.IN**
Denumirea fișierului de ieșire: **ATLANTIS.OUT**

Descrierea problemei

După ce a reușit să treacă de poarta cetății *Atlantis*, *Iahim Uratrox*, marele explorator al *Atlantidei*, a descoperit o mulțime dintre vestigiile vechii civilizații.

Singura clădire în care nu a putut intra este palatul foștilor suverani atlanți. Se presupune că în interiorul acestui palat se pot găsi documentele care să explice motivul scufundării miticului oraș. Din nefericire, intrarea în palat este protejată și ea de un cifru.

Mecanismul este unul mult mai simplu, el constând din doar două dispozitive de introducere a codului. Pe mecanism este gravat, la fel ca și la intrare, un număr format din șase cifre.

Bineînțeles, savanții au reușit imediat să descopere algoritmul de funcționare al mecanismului. Cele două dispozitive corespund semnelor `'+'` și `'-'`. Fiecare atingere a unui dispozitiv duce la adunarea sau scăderea unui număr. Acest număr variază în funcție de numărul de ordine al atingerii. La a i-a atingere, numărul care se adună sau se scade este i^2. Se pornește cu valoarea 0.

Așadar, la fiecare atingere i valoarea corespunzătoare mecanismului scade sau crește cu i^2. În final, pentru a deschide poarta, trebuie să se obțină valoarea gravată pe mecanism.

Date de intrare

Fișierul de intrare conține o singură linie pe care se află cele șase cifre ale numărului gravat pe mecanismul cifrului, neseparate prin spații. Există posibilitatea ca prima, primele două, primele trei, primele patru sau primele cinci cifre să fie 0. Totuși, cel puțin una dintre cele șase cifre este diferită de zero.

Date de ieșire

Fișierul de ieșire va conține o singură linie pe care se vor semnele corespunzătoare atingerilor mecanismului. Al i-lea semn de pe această linie va corespunde celei de-a i-a atingeri. Dacă prin această atingere se adună valoarea i^2, atunci semnul va fi `'+'`. Dacă atingerea duce la scăderea valorii i^2, atunci semnul va fi `'-'`.

Restricții și precizări

- Valoarea numărului inscripționat pe mecanismul cifrului este cuprinsă între 1 și 999999.
- În cazul în care nu poate fi descoperită o secvență de atingeri, *Iahim Uratrox* nu va mai putea deschide poarta folosind o încărcătură explozivă așa cum ar fi putut face atunci când se afla în fața porții orașului, deoarece ar risca să distrugă prea multe vestigii importante; astfel, cel mai mare secret al *Atlantidei* va fi păstrat pentru totdeauna. Din fericire, după câteva luni, exploratorul s-a întors și a prezentat motivele scufundării *Continentului Pierdut*; se pare că explicația era atât de îngrijorătoare încât s-a decis ca motivele să nu fie cunoscute publicului. Totuși, concluzia evidentă care poate fi trasă este că există întotdeauna o secvență corectă de atingeri care duc la deschiderea porții.
- Poarta se deschide chiar dacă numărul atingerilor nu este minim.
- Dacă există mai multe soluții, doar una trebuie scrisă în fișierul de ieșire.

Exemple

ATLANTIS.IN	ATLANTIS.OUT
000012	++-+

ATLANTIS.IN	ATLANTIS.OUT
000055	+++++

Explicații

- $0 + 1^2 + 2^2 - 3^2 + 4^2 = 0 + 1 + 4 - 9 + 16 = 12$;
- $0 + 1^2 + 2^2 + 3^2 + 4^2 + 5^2 = 0 + 1 + 4 + 9 + 16 + 25 = 55$.

Timp de execuție/test: **5 secunde**

Runda 5, problema 3

TIC-TAC-TOE

Denumirea subdirectorului: **TICTAC**

Denumirea fișierului de intrare: -

Denumirea fișierului de ieșire: -

Descrierea problemei

Se consideră o tablă de dimensiuni 3 × 3 care este inițial goală. Pe rând, jucătorii mută, marcând una din căsuțele libere cu un semn propriu. Prin convenție semnul primului jucător este `"X"`, iar semnul celui de-al doilea este `"0"`. Jucătorul care obține primul trei semne proprii pe aceeași linie, coloană sau diagonală este declarat câștigător. Dacă toate căsuțele au fost marcate și nici un jucător nu a câștigat atunci se declară *remiză*.

Presupunând că primul jucător este reprezentat de calculator, se cere să se programeze o strategie pentru al doilea jucător. Calculatorul va realiza întotdeauna operații deterministe pentru a efectua o mutare.

Pentru fiecare partidă câștigată veți primi 0.002 puncte. Pentru fiecare remiză veți primi 0.001 puncte. Pentru o partidă pierdută nu veți primi nici un punct.

Va trebui ca, într-un interval de o secundă, să jucați cât mai multe partide. Timpul consumat de calculator pentru a determina pozițiile vă permite (în cazul în care veți folosi un algoritm eficient) să jucați cel puțin 100000 de partide.

Cele nouă căsuțe ale tablei vor fi numerotate astfel:

0	1	2
3	4	5
6	7	8

Programatorii în *Pascal* vor avea la dispoziție un *unit* numit **TICTAC.PAS**, care va conține două proceduri și o funcție.

La începutul jocului va trebui, obligatoriu, să fie apelată procedura `InitGame`, care nu are nici un parametru și este folosită pentru inițializarea jocului.

Pentru a efectua o mutare veți apela procedura `MyMove`. Aceasta are un singur parametru care indică poziția în care veți amplasa un `"0"`.

Pentru a afla mutarea efectuată de adversar veți apela funcția `YourMove`. Aceasta nu are nici un parametru și returnează poziția în care calculatorul a amplasat un `"X"`. În cazul în care jocul se încheie, funcția va returna valoarea `-1`.

Programatorii în C/C++ vor avea la dispoziție un *header* numit **TICTAC.H**, care va conține trei funcții.

La începutul jocului va trebui, obligatoriu, să fie apelată funcția `InitGame`, care nu are nici un parametru și este folosită pentru inițializarea jocului.

Pentru a efectua o mutare veți apela funcția `MyMove`. Aceasta are un singur parametru care indică poziția în care veți amplasa un `"0"`.

Pentru a afla mutarea efectuată de adversar veți apela funcția `YourMove`. Aceasta nu are nici un parametru și returnează poziția în care calculatorul a amplasat un `"X"`. În cazul în care jocul se încheie, funcția va returna valoarea `-1`. Desfășurarea unui joc este următoarea: la început se va inițializa jocul, după care se va cere efectuarea unei mutări de către adversar. Apoi, alternativ, se vor efectua mutări și se vor cere mutările adversarului până în momentul în care funcția `YourMove` va returna valoarea `-1`. Chiar dacă mutarea voastră este câștigătoare, va trebui să mai apelați o dată funcția `YourMove` care va returna, obligatoriu, valoarea `-1`. Următorul joc va începe după o nouă inițializare.

Funcția `YourMove` va întrerupe execuția programului în momentul în care va expira timpul alocat partidelor.

În cazul în care nu veți respecta secvența de mutări, veți pierde jocul respectiv. Această sitiuație poate apărea dacă:

- se efectuează două mutări consecutive;
- se cere efectuarea a două mutări consecutive;
- se efectuează sau se cer mutări fără a se inițializa jocul;
- se inițializează jocul înaintea terminării unui joc (înainte ca funcția `YourMove` să returneze valoarea `-1`);
- nu se cere efectuarea primei mutări de către advesar;
- orice altă situație în care nu se respectă regulile jocului.

Câte o variantă a *unit*-ului și a *header*-ului sunt disponibile pentru *download* la adresa **www.ginfo.ro/concurs/tictactoe.shtml**. Calculatorul va folosi o strategie aleatoare, deci acestea nu vor fi folosite în timpul evaluării.

Programul vostru nu va trebui să citească date din nici un fișier și nu va trebui să scrie date în nici un fișier.

Modelarea EXPRESIILOR folosind tehnici EVOLUTIVE

Mihai Oltean, Crina Groşan

Rezolvarea problemelor care implică folosirea unor expresii matematice a fost intens studiată în ultimele decenii. În acest articol este prezentată o tehnică evolutivă pentru rezolvarea de probleme dificile care conțin expresii matematice.

Algoritmi evolutivi

Rezolvarea problemelor folosind metode evolutive este un domeniu de tradiție. El își are rădăcinile în anii '50, când mai mulți cercetători au preluat metafora evolutivă din natură, aplicând-o la rezolvarea problemelor din diverse domenii. Metafora evolutivă este transpusă, informatic, într-un număr foarte mare de algoritmi. Totuși, câteva considerente generale pot fi indicate.

Se lucrează cu o populație (vector, listă) de posibile soluții. Acestea se mai numesc uneori și *indivizi*, *cromozomi* etc. Indivizii populației inițiale sunt generați aleator. Cu ajutorul selecției și al operatorilor genetici indivizii evoluează și, eventual, vor converge către soluție.

Indivizii sunt evaluați pe baza unei așa-zise funcții de *fitness* (calitate). *Fitness* înseamnă speranță de viață. Cu cât un individ este mai performant, cu atât speranța lui de viață este mai mare.

Asupra indivizilor din populație se aplică operatori genetici. Acești operatori sunt dependenți de reprezentare. Dintre cei mai răspândiți operatori genetici amintim *încrucișarea* și *mutația*. Prin încrucișare, de obicei, doi părinți dau naștere a doi fii. Prin mutație se aplică o mică "perturbație" a valorii unui individ.

De obicei, nu toți indivizii sunt aleși pentru aplicarea operatorilor genetici. Modul și ordinea în care se aplică operatorii genetici asupra indivizilor variază foarte mult. De exemplu, într-un algoritm genetic din populație sunt selectați indivizi în funcție de *fitness*-ul lor. Indivizii selectați sunt introduși într-o populație intermediară. Indivizii din această populație intermediară sunt supuși operatorilor genetici de încrucișare și mutație. Indivizii rezultați în urma acestor operații constituie "următoarea generație", adică o nouă populație care va fi din nou supusă selecției și operatorilor genetici. Acest proces va continua pentru un anumit număr de generații.

Programare genetică

Programarea genetică este o ramură a algoritmilor evolutivi care se ocupă cu evoluarea codurilor sursă.

Se dorește, de exemplu, evoluarea unui program *Pascal* sau *C++* care să rezolve problema drumului *hamiltonian* sau problema ieșirii dintr-un labirint. Să presupunem că nu știm cum să rezolvăm această problemă. Așadar, nu ne interesează să obținem o soluție pentru un set oarecare de date, ci dorim să obținem un program sursă care să genereze o soluție cât mai aproape de cea corectă pentru orice date de intrare. Cu alte cuvinte, ne interesează să obținem ca rezultat un program asemănător cu cel pe care l-am fi putut scrie dacă am fi știut rezolva problema.

Din punct de vedere evolutiv, abordarea unor astfel de probleme se face generând o mulțime (populație) aleatoare de coduri sursă care apoi sunt selectate pe baza funcției de *fitness* și sunt evoluate cu ajutorul unor operatori genetici specifici.

În primul rând, trebuie să atribuim o funcție de calitate (funcția *fitness*) fiecărui program generat. Această funcție de *fitness* trebuie să reflecte performanțele programului căruia îi este atașată.

De obicei, atașarea unei funcții de *fitness* se face rulând programul respectiv și măsurând calitatea soluției în raport cu o soluție care se cunoaște a fi optimă. Un program va avea o calitate mai mare, dacă soluția generată va fi mai apropiată de soluția optimă. Dacă nu se cunoaște o soluție optimă, atunci această situație nu implică apariția unor dificultăți suplimentare, deoarece noi dorim să maximizăm sau să minimizăm valoarea *fitness*-ului.

Evoluarea programelor sursă se realizează prin intermediul operatorilor genetici specifici. De exemplu, un operator de recombinare poate însemna alipirea unor secvențe dintr-un cod sursă cu secvențe din alt cod sursă. Un operator de mutație ar putea însemna inserarea de noi instrucțiuni în codul sursă. De notat este faptul că alegerea operatorilor folosiți este la latitudinea programatorului. Totul depinde de imaginația celui care propune o metodă. Multe articole științifice prezintă noi operatori genetici cu performanțe superioare celor existenți.

Evident, în urma aplicării acestor operatori genetici pot fi generate coduri sursă care conțin greșeli de sintaxă. De asemenea, sunt generate secvențe de cod sursă nefolositoare. Un exemplu în acest sens este secvența de instrucțiuni:

```
i := i + 1;
i := i - 1;
```

După executarea acestor instrucțiuni valoarea variabilei `i` rămâne neschimbată, deci uneori ar fi utilă eliminarea unora dintre aceste secvențe. Eliminarea nu este recomandată întotdeauna pentru că ulterior o altă aplicare a operatorilor genetici ar putea insera și alte instrucțiuni între cele două de mai sus, ducând astfel la eliminarea redundanțelor.

Evoluarea programelor sursă în limbajele *Pascal* sau *C* este o muncă dificilă și rezultatele sunt foarte puțin spectaculoase. Obținerea prin evoluție a unui cod sursă *Pascal* sau *C* pentru rezolvarea problemei drumului *hamiltonian* este un fel de *Fata Morgana*. *John Koza*, unul dintre pionierii domeniului, a declarat că acasă are o rețea cu 500 de calculatoare cu ajutorul căreia evoluează coduri sursă complexe. Și nici în aceste condiții nu a obținut rezultate semnificative.

De obicei, se lucrează cu reprezentări mai simple ale programelor și anume cu arbori. Generarea unui arbore atașat unui program nu este o muncă dificilă. În plus, există limbaje de programare (de exemplu *LISP*) în care programele sunt scrise sub forma unor liste ușor transformabile în arbori.

Expresii aritmetice

Tot în domeniul programării genetice intră și generarea unor funcții matematice care îndeplinesc anumite criterii. Se dorește găsirea unei funcții `f` care satisface cât mai bine o mulțime de perechi `(x, f(x))` date.

Ne propunem să lucrăm cu expresii aritmetice. Problema care apare este cum reprezentăm eficient indivizii astfel încât în urma aplicării operatorilor genetici să rezulte expresii aritmetice valide. De exemplu, considerând expresiile aritmetice:

E_1 = `(a + b - d) * c - d * a`

E_2 = `x * (c + d * (a - b))`

și considerând operatorul genetic clasic de recombinare ca fiind recombinarea cu un singur punct de tăietură, obținem expresiile:

F_1 = `(a(c + d * (a-b))` și

F_2 = `x * + b - d) * c - d * a`.

Aici punctul de tăietură a fost ales după poziția a doua. Se observă că cele două expresii aritmetice rezultate nu sunt valide.

Aceleași dificultăți se întâlnesc și în cazul operatorului de mutație.

O soluție ar fi căutarea unor operatori genetici noi care să poată fi aplicați asupra expresiilor aritmetice. Aceștia ar trebui să fie destul de puternici pentru a efectua o căutare completă în spațiul posibilelor soluții și suficient de rapizi pentru ca rezolvarea problemei să fie obținută într-un timp scurt. O modalitate ar fi reprezentarea sub formă de arbori a expresiilor aritmetice. Acest mod de reprezentare este specific *programării genetice*.

Arborii sunt structuri complexe de reprezentare a datelor. Operatorul genetic de încrucișare folosit pentru evoluarea arborilor constă, de obicei, în schimbul reciproc de subarbori.

Operatorul de mutație modifică un arbore prin adăugarea sau ștergerea unor subarbori. În urma aplicării operatorilor genetici pe arbori vor rezulta arbori valizi, care generează expresii aritmetice corecte.

Această metodă prezintă dificultăți datorate, în principal, aplicării greoaie a operatorilor genetici asupra arborilor (schimbarea între ei a doi subarbori este o sarcină destul de dificilă).

O altă problemă care apare când se lucrează cu reprezentări arborescente este dimensiunea arborilor. Fără un control strict al dimensiunii, se poate întâmpla ca arborii rezultați să crească excesiv, lucru care duce la o manipulare greoaie și, mai ales, la ocuparea completă a memoriei.

Gene expression programming

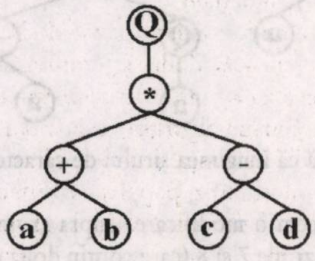
O altă soluție ar fi căutarea unor reprezentări mai eficiente pentru expresiile aritmetice. O astfel de reprezentare este prezentată în cele ce urmează. Acest tip de codificare a unei expresii aritmetice apare în [1].

De obicei, când se lucra cu expresii aritmetice, cele mai des folosite parcurgeri erau traversările în inordine, postordine si preordine. Codificarea propusă de *Ferreira* este o extensie a parcurgerii pe niveluri a unui arbore.

Fie expresia matematică:

$$F = \sqrt{(a+b)\times(c-d)}.$$

Reprezentarea sub formă de arbore a acestei expresii este:



Aici "`Q`" simbolizează rădăcina pătrată. Parcurgerea pe niveluri a acestui arbore generează șirul: `Q*+-abcd`.

În cele ce urmează vom vedea care sunt avantajele acestei reprezentări atunci când expresiile matematice sunt evoluate cu ajutorul unor mecanisme genetice.

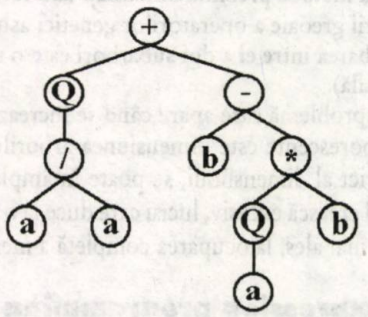
Scopul nostru este de a obține întotdeauna expresii matematice valide. De aceea vom impune câteva restricții asupra reprezentării și operatorilor.

Un cromozom este alcătuit dintr-un cap și o coadă. Capul conține simboluri care reprezintă atât operanzi cât și operatori. Coada conține simboluri care reprezintă doar operanzi. Pentru fiecare problemă în parte este aleasă o valoare `h` pentru lungimea capului. Lungimea cozii `t` este calculată în funcție de lungimea capului și a numărului de argumente `n` al operatorului cu cele mai multe argumente, astfel: `t = (n - 1) * h + 1`.

Să considerăm un cromozom care conține operatorii `{Q, +, -, *, /}` și operanzii `{a, b}`. În acest caz `n = 2`. Pentru lungimea capului cromozomului `h = 10`, rezultă că lungimea cozii este `t = 11`. Un posibil cromozom care verifică aceste cerințe este următorul (coada este reprezentată folosind caractere îngroșate):

`+Q-/b*aaQ`**`baabaabbaaab`**

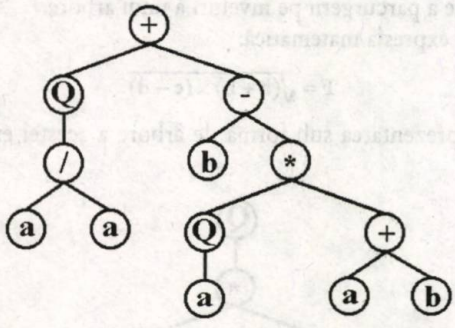
Acest șir codifică arborele:



Se observă că, deși cromozomul are lungimea 21, doar primele 10 caractere codifică arborele reprezentat mai sus. Restul de caractere sunt nefolosite.

Să presupunem că o mutație apare în poziția 10, schimbând caraterul "`b`" în "`+`". Este obținut următorul cromozom: `+Q-/b*aaQ`**`aabaabbaaab`**

În acest caz, arborele codificat de acest cromozom este:

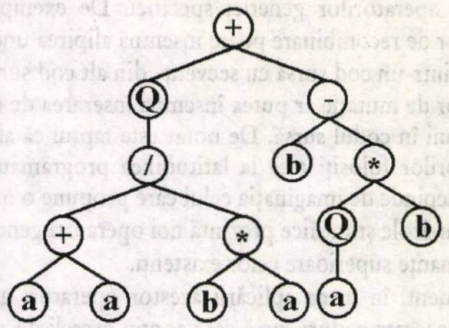


Se observă că lungimea șirului de caractere util este în acest caz 12.

Să mai facem o modificare asupra cromozomului. Să schimbăm pozițiile 7 și 8 (care conțin două caractere "`a`") în "`+`" respectiv "`*`". Se obține următorul cromozom:

`+Q-/b*+*Q+`**`aabaabbaaab`**

Arborele codificat de acest cromozom este:

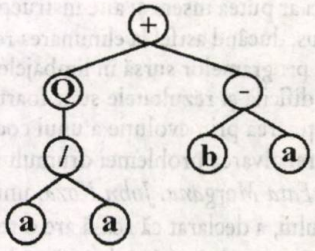


Se observă că pentru acest arbore numărul caracterelor folosite din cromozom este mult mai mare.

Până în acest moment am făcut doar mutații care creșteau dimensiunea arborelui; este posibilă și situația opusă care duce la scăderea dimensiunii. Dacă pe poziția 6 din arbore schimbăm caracterul "`*`" cu "`a`" obținem următorul cromozom:

`+Q-/ba+*Q+`**`aabaabbaaab`**

Acest cromozom codifică arborele:



Se observă că arborele obținut are cu trei noduri mai puțin decât arborele inițial. Chiar dacă fiecare cromozom are o lungime fixă, acesta poate codifica arbori de o mare varietate, de la cel mai simplu care conține doar un nod (în acest caz pe prima poziție a cromozomului se află un operand), până la cel mai complex care valorifică toate caracterele cromozomului (în acest caz toate elementele capului sunt operatori de aritate maximă).

Din exemplele anterioare se observă că, oricât de profunde ar fi modificările într-un cromozom, acesta va codifica întotdeauna un arbore valid. Pentru ca acest lucru să fie adevărat trebuie ca structura cromozomului să rămână nealterată, adică să nu se permită inserarea în coadă a unor operatori.

Decodificarea informației

Decodificarea informației înseamnă trecerea de la un cromozom la arborele corespunzător. În cromozom arborele este codificat sub forma rezultată în urma parcurgerii pe niveluri. Această reprezentare este mai greu de manipulat atunci când se dorește evaluarea arborelui. Dacă avem un arbore binar care codifică o expresie aritmetică, atunci pentru evaluarea ei trebuie să parcurgem arborele în postordine. De aceea trebuie realizată trecerea de la forma conținută în cromozom la o altă formă ușor utilizabilă pentru parcurgerea în postordine.

De exemplu, dacă arborele codificat în cromozom este binar, atunci se poate construi ușor o reprezentare în *Heap* a acestuia. O posibilitate este alegerea unei reprezentări statice: se folosește un tablou cu trei coloane și N linii (N este numărul de noduri din arbore), în care pe fiecare linie în prima coloană s-a reținut informația atașată unui nod, iar pe celelalte două poziții s-au reținut pozițiile din care încep subarborii stâng și drept a nodului curent.

Translatarea începe întotdeauna de la prima poziție din cromozom și se continuă atât timp cât arborele care se construiește nu devine complet. Restricțiile impuse cromozomului garantează întotdeauna că dintr-un cromozom se poate construi un arbore valid.

Fitness-ul unui cromozom

Se evaluează expresia codificată într-un cromozom. Cu cât valoarea rezultată este mai aproape de rezultatul dorit, cu atât *fitness*-ul cromozomului este mai mare.

Vom nota cu f funcția de *fitness*. Aceasta este definită pe mulțimea posibilă a cromozomilor și are, de obicei, valori în mulțimea numerelor reale pozitive.

Selecția

Pentru selecție vom folosi metoda turnirului binar. Aceasta constă în selectarea aleatoare a doi indivizi și compararea lor directă. Cel mai performant dintre aceștia este inserat în populația intermediară, care va constitui rezerva de indivizi pentru încrucișare și mutație. Populația intermediară se mai numește și *piscină de încrucișare*.

Încrucișarea

Prin încrucișare, din doi indivizi numiți părinți, se obțin doi fii. Deoarece indivizii sunt șiruri de caractere, se poate folosi încrucișarea cu un singur punct de tăietură. Spre exemplu să considerăm doi cromozomi:

$x = x_1\ x_2 \dots x_k\ x_{k+1} \dots x_r$

și

$y = y_1\ y_2 \dots y_k\ y_{k+1} \dots y_r$.

În urma recombinării (încrucișării) se schimbă între cei doi cromozomi secvențele aflate la dreapta punctului de tăietură k. Cromozomii fii vor fi:

$x' = x_1\ x_2 \dots x_k\ y_{k+1} \dots y_r$

și

$y' = y_1\ y_2 \dots y_k\ x_{k+1} \dots x_r$.

Mutația

Operatorul de *mutație* constă în înlocuirea unui simbol cu un alt simbol. În secvența care alcătuiește capul cromozomului, fiecare simbol poate fi transformat în orice alt simbol. În secvența care alcătuiește coada cromozomului, doar operanzii pot fi transformați în alți operanzi.

De obicei, nu toate simbolurile din cromozom sunt supuse mutației. De fapt, mutația este aplicată asupra unui număr relativ mic de simboluri din cromozom.

Se consideră o probabilitate de mutație p_m pe baza căreia sunt schimbate simbolurile din cromozom. Algoritmul folosit pentru realizarea mutațiilor este descris în continuare:

Pentru fiecare poziție a cromozomului se execută:

P_1. Se generează un număr aleator q în intervalul [0, 1].

P_2. Dacă $q < p_m$, atunci se execută mutația poziției respective, schimbând un simbol cu un altul; în caz contrar ($q \geq p_m$), poziția respectivă nu se schimbă.

Problema 1: Expresie de valoare dată

Se dau constantele întregi C_1, C_2, ..., C_N, S. Se cere să se construiască o expresie aritmetică E care conține operatorii {+, -, *} și eventual paranteze. Operanzii admiși sunt constantele C_1, C_2, ..., C_N. Fiecare constantă poate apărea o dată, de mai multe ori, sau niciodată în construcția expresiei E. Valoarea expresiei E trebuie să fie cât mai apropiată de S.

Datele de intrare se citesc dintr-un fișier text având următoarea structură:

- pe prima linie sunt scrise valorile N și S,
- pe următoarea linie sunt scrise constantele C_1, C_2, ..., C_N, despărțite prin spații.

Rezolvare

Pentru rezolvarea acestei probleme vom folosi un algoritm genetic standard. Cromozomii reprezintă posibilele soluții ale problemei și sunt codificați în forma descrisă anterior. Pentru simplificare cromozomii sunt vectori de valori întregi din intervalul [0, N + 3]. Numerele 0, 1, 2 codifică operatorii +, -, respectiv *, iar numerele de la 3 la N + 3 reprezintă indicii constantelor C_1, C_2, ..., C_N (numărul 3 corespunde constantei C_1, numărul 4 corespunde constantei C_2 etc.).

Lungimea capului cromozomilor este un parametru al algoritmului și poate fi fixat de către utilizator. De asemenea, parametri ai algoritmului mai sunt și dimensiunea populației, numărul de iterații și numărul celor mai buni indivizi care sunt trecuți automat în populația următoare (Nr_Elitism). De obicei, un număr de 100 de iterații și un număr de 100 de indivizi într-o populație sunt suficienți.

Codul sursă poate fi descărcat de la următoarea adresă: **http://www.ginfo.ro/revista/12_2/index.shtml.**

Exemplu

Intrare

```
10 4
3 5 2 6
```

Ieșire

```
*2563
```

Șirul furnizat la ieșire corespunde expresiei 2*5.

Discuție

Se poate lucra și cu numere reale. În acest caz se poate introduce și operatorul /.

Se pot introduce restricții asupra folosirii operanzilor. De exemplu, se poate introduce restricția ca fiecare ope-

rand să se folosească o singură dată. Această restricție va afecta puțin felul în care sunt codificate expresiile.

Cei care doresc pot implementa o variantă a algoritmului genetic, folosind însă codificarea sub formă de arbori binari alocați dinamic. Se va observa diferența clară între viteza de execuție a acestui program și viteza de execuție și cantitatea de memorie alocată de programele care folosesc direct codificarea sub formă de arbore.

Problema 2: Regresie simbolică

Fie o funcție `f` care depinde de un singur parametru `x`. Dându-se o mulțime de perechi $S = \{(x_i, f(x_i)), i = 1, \ldots, N\}$, se cere să se determine o funcție `f` care satisface cât mai bine aceste restricții. Funcția nu poate să conțină constante.

Datele de intrare se citesc dintr-un fișier text având următoarea structură:

- pe prima linie se află numărul `N` al perechilor `(x, f(x))`;
- pe fiecare dintre următoarele `N` linii se află câte o pereche sub forma `x f(x)`.

Rezolvare

În primul rând definim mulțimea operatorilor pe care dorim să îi folosim. Aceștia pot fi cei aritmetici `(+, -, *, /)`, dar și alte funcții matematice `(sin, cos, tg, log, sqrt` etc.). Deci, într-o expresie vom întâlni operatorii definiți și operandul `x`. Astfel, structura cromozomului va fi asemănătoare cu cea de la problema precedentă.

Diferența care apare este la calcularea funcției de *fitness*. Dacă la problema precedentă calitatea unei soluții era dată de diferența între valoarea ei și valoarea țintă `S`, aici problema este puțin mai complicată, deoarece pentru diferite valori ale lui `x` vom avea diferite valori țintă `f(x)`. O variantă posibilă ar fi să însumăm (pentru fiecare valoare x_i) diferențele între valoarea obținută și valoarea țintă.

Discuție

Restricția privitoare la apariția constantelor într-o expresie va fi discutată în problema următoare.

Deoarece se lucrează cu numere reale, se cere ca expresia rezultată să se evalueze la valori apropiate cu o precizie dată față de valorile țintă.

Problema 3: Ghicește următorul număr

Se dau primii `M` termeni din șirul de constante $(a_N)_{N\geq 0}$. Acest șir este alcătuit pe baza unei reguli. Regula este o expresie de un singur parametru `(N)`. Se cere să se determine următorul număr din secvență (a_{M+1}) și eventual regula după care a fost alcătuit.

Exemplu

Pentru secvența `1, 15, 129, 547, 1593, 3711, 7465, 13539, 22737, 35983` algoritmul trebuie să ghicească următorul număr ca fiind `54321`.

Șirul este construit conform regulii: $a_N = 5N^4 + 4N^3 + 3N^2 + 2N + 1$.

Rezolvare

Această problemă este un caz particular al problemei precedente, deoarece domeniul variabilei `N` este acel al numerelor întregi nenegative. Totuși, problema este și mai complicată deoarece implică utilizarea de constante.

Rezolvarea este asemănătoare cu soluțiile propuse la problemele precedente. Lângă fiecare apariție a variabilei `N` se mai adaugă și o constantă care va evolua cu ajutorul operatorilor genetici. Se poate folosi mutația ca singurul operator care evoluează constantele. În acest caz, mutația unei constante înseamnă adăugarea sau scăderea unei mici valori la aceea constantă.

Probleme propuse

În cele ce urmează vom propune spre rezolvare câteva probleme ale căror soluții sunt bazate pe tehnicile prezentate în cadrul acestui articol.

Găsește recurența

Se dau primii `M` termeni ai unei recurențe $a_N = a_1 * a_{N-k} + a_2 * a_{N-k-1} + \ldots + a_p * a_{N-k-p} + 1$. Se cere să se determine relația de recurență care satisface acest șir. Termenii a_i sunt numere reale.

Funcții trigonometrice

Se consideră o expresie care conține funcțiile trigonometrice `sin`, `cos`, `tg` și `ctg`, precum și diferite alte funcții aritmetice. Se cere să se determine alte expresii echivalente cu cea dată.

Teoreme

Se consideră anumite valori pentru catetele și ipotenuza unui triunghi dreptunghic. Să se determine expresia teoremei lui *Pitagora* folosind tehnici evolutive.

Să se aplice aceeași metodă pentru determinarea altor expresii pentru diferite teoreme din fizică și matematică.

Bibliografie

[1] Cândida Ferreira, *Gene Expression Programming: a New Adaptive Algorithm for Solving Problems, Complex Systems* (în curs de apariție).

[2] Mihai Oltean, *Proiectarea și Implementarea Algoritmilor*, Editura Computer Libris Agora, Cluj-Napoca, 2000.

[3] D. Dumitrescu, *Algoritmi genetici și strategii evolutive*, Editura Albastră, Grupul MicroInformatica, Cluj-Napoca, 2000.

[4] J. Koza, *Genetic Programming: On the programming of Computers by Means of Natural Selection*, Cambridge, MA: MIT Press, 1992.

[5] Garey Johnson, *Computers and Intractability: A Guide to NP-Completeness*, Freeman & Co, San Francisco

[6] R. Motwani, *P. Raghavan, Randomized Algorithms*, Cambridge University Press, 1995.

Mihai Oltean și Crina Groșan sunt masteri și doctoranzi ai Universității Babeș-Bolyai din Cluj-Napoca. Pot fi contactați prin e-mail la adresele **moltean@cs.ubbcluj.ro**, *respectiv* **cgrosan@cs.ubbcluj.ro**.

MODELE pentru INTERCONECTARE

Mihai Scorțaru

Noțiunea de rețea de calculatoare a apărut în deceniul 8 al secolului XX; dezvoltarea ulterioară a industriei calculatoarelor a dus la modificări importante ale semnificației acestei noțiuni. În acest articol vom încerca să prezentăm câteva noțiuni introductive referitoare la acest subiect.

Introducere

La început, prin termenul de rețea de calculatoare se înțelegea conectarea mai multor terminale la un calculator puternic. De obicei, terminalele nu aveau putere de calcul, toate operațiile fiind efectuate de către calculatorul central.

Pe măsură ce importanța stațiilor de lucru a crescut, semnificația noțiunii s-a modificat. Acum, prin rețea de calculatoare se înțelege o mulțime de stații de lucru care au, fiecare, o anumită putere de lucru și o anumită capacitate de memorare, și care sunt conectate între ele.

Principalele caracteristici ale unei rețele de calculatoare se referă la *tipul*, *localizarea* și *modul de accesare* al diverselor ***resurse partajate***.

În funcție de localizarea componentelor, rețelele de calculatoare pot fi clasificate astfel:

- ***rețele locale*** (*LAN - Local Area Network*) - distanțele între stații sunt, cel mult, de ordinul kilometrilor;
- ***rețele metropolitane*** (*MAN - Metropolitan Area Network*) - distanțele între stații pot ajunge la zeci de kilometri;
- ***rețele cu întindere geografică mare*** (*WAN - Wide Area Network*) - distanțele pot fi de ordinul miilor de kilometri; suprafața acoperită de acest tip de rețele corespunde unei țări sau chiar întregului *Glob*.

Standarde

Datorită domeniului foarte larg de utilizare al rețelelor de calculatoare apare necesitatea stabilirii unor standarde pentru construirea și funcționarea acestora. Din nefericire, problema standardizării este una foarte delicată datorită evoluției rapide înregistrate în acest domeniu. În prezent există standarde folosite doar de anumite firme care au furnizat produse de acest tip la nivel internațional, standarde ***de facto***, care sunt folosite de producători, dar nu sunt recunoscute de organizațiile de standardizare, standarde ***de jure***, care sunt elaborate de diferite organizații, dar majoritatea au o răspândire mică și tehnologii noi care se află în curs de standardizare și sunt promovate de diverse companii.

Există două mari categorii de organizații de standardizare în domeniul rețelelor de calculatoare: ***grupurile*** de standardizare și ***forumurile*** pentru dezvoltarea aplicațiilor industriale.

Cele mai importante grupuri de standardizare sunt *ITU* (*International Telecommunications Union*), *ANSI* (*American National Standards Institute*), *IEEE* (*Institute for Electrical and Electronics Engineers*), *ETSI* (*European Telecommunications Standards Institute*), *Bellcore* (*Bell Communications Research*) și *ISO/IEC* (*International Standards Organization / International Electrotechnical Commision*).

Forumurile cu o influență majoră în standardizare sunt *Frame Relay Forum*, *SMDS Interest Group*, *ATM Forum* și *Internet Engineering Task Force*.

Comunicația datelor

Unul dintre cele mai importante concepte în domeniul rețelelor de calculatoare este reprezentat de comunicația datelor. Există trei noțiuni importante care se referă la comunicații:

- ***date*** - reprezintă informațiile propriu-zise;
- ***semnale*** - sunt purtătoarele datelor;
- ***transmisii*** - se referă la comunicarea datelor prin propagarea și procesarea semnalelor.

Alți doi termeni importanți sunt ***analog*** și ***digital***. Prin combinarea termenilor se obțin câteva expresii importante:

- ***date analogice*** - pot avea orice valoarea cuprinsă într-un anumit interval;
- ***date digitale*** - pot avea valori discrete cuprinse într-o mulțime finită;
- ***semnale analogice*** - unde electromagnetice continue;
- ***semnale digitale (discrete)*** - sunt secvențe de impulsuri de curent sau tensiune.

Principalele tehnici de transmitere a datelor sunt:

- *transmisia digitală a datelor digitale* - folosește tehnici de codificare a datelor;
- *transmisia digitală a datelor analogice* - semnalul analogic este aproximat printr-un semnal digital;
- *transmisia analogică* - se folosește modularea și demodularea semnalelor care reprezintă informația.

Transmisia digitală a datelor digitale

Cea mai simplă modalitate pentru reprezentarea datelor binare este folosirea a două niveluri de tensiune. Pentru valoarea `0` se va folosi un nivel scăzut de tensiune, iar pentru valoarea `1` se va folosi un nivel ridicat. Această tehnică poartă denumirea *NRZ* (*Non Return to Zero*) sau *NRZ-L* (*Non Return to Zero Level*).

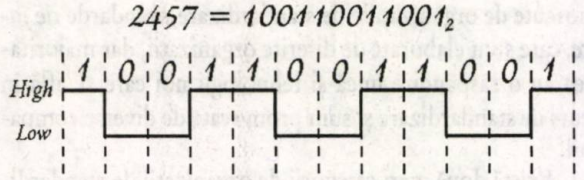


Fig. 1: Codificarea Non Return to Zero

O altă metodă de reprezentare a datelor este *NRZI* (*No Return to Zero, Invert on Ones*). Valoarea `1` este codificată prin existența unei tranziții (variație de tensiune) la începutul perioadei de timp corespunzătoare unui bit, iar valoarea `0` este codificată prin inexistența acestei tranziții.

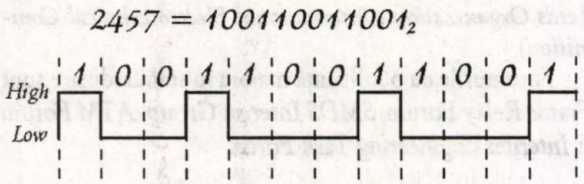


Fig. 2: Codificarea Non Return to Zero, Invert on Ones

O variantă a acestei tehnici este metoda ***Manchester***. În acest caz, la mijlocul perioadei corespunzătoare unui bit, există întotdeauna o tranziție. Sensul tranziției indică valoarea corespunzătoare bitului: o creștere de tensiune corespunde valorii `0`, iar o scădere de tensiune corespunde valorii `1`. Tranziția de la mijlocul perioadei corespunzătoare unui bit poate fi folosită și ca semnal de ceas.

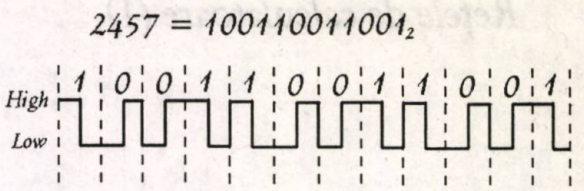


Fig. 3: Codificarea cu metoda Manchester

O combinație a tehnicilor *NRZI* și *Manchester* este metoda *Manchester diferențiat*. Tranziția de la mijlocul perioadei corespunzătoare unui bit este folosită ca semnal de ceas, iar datele sunt indicate de prezența (valoarea `0`) sau lipsa (valoarea `1`) unei tranziții la începutul perioadei.

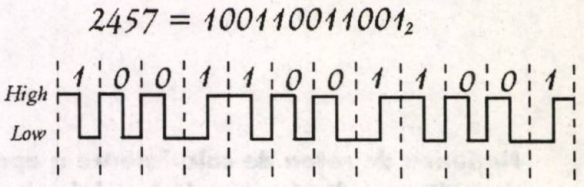


Fig. 4: Codificarea cu metoda Manchester diferențial

Transmisia analogică a datelor digitale

Această tehnică de transmitere a datelor se bazează pe folosirea unui semnal de frecvență constantă numit purtătoare de date. Codificarea datelor se realizează prin modificarea unuia dintre cei trei parametri ai semnalului: amplitudine, frecvență și fază.

- codificarea prin amplitudine implică folosirea a două valori diferite pentru amplitudinea semnalului pentru cele două valori binare; tehnica este cunoscută sub denumirea *ASK* (*Amplitude Shift Keying*).

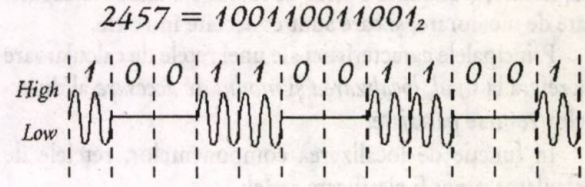


Fig. 5: Codificarea prin modelare în amplitudine

- codificarea prin frecvență implică folosirea a două valori diferite pentru frecvența semnalului pentru cele două valori binare; tehnica este cunoscută sub denumirea *FSK* (*Frequency Shift Keying*).

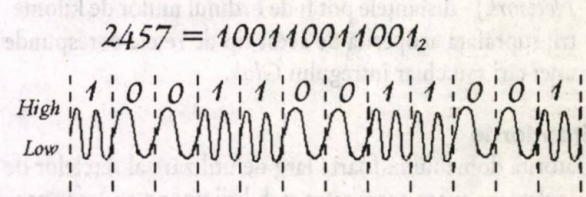


Fig. 6: Codificarea prin modelare în frecvență

- codificarea prin deplasarea fazei implică folosirea unor semnale rafală pentru cele două valori binare; valoarea `0`

este codificată printr-un semnal cu aceeași fază cu semnalul anterior, iar valoarea 1 este codificată printr-un semnal cu fază opusă; tehnica este cunoscută sub denumirea *PSK* (*Phase Shift Keying*).

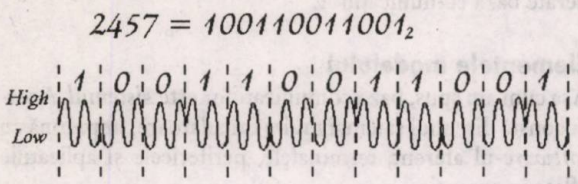


Fig. 7: Codificarea prin modelare în fază

Transmisia digitală a datelor analogice

Modalitatea digitală de transmitere a datelor analogice se bazează pe teorema eșantionării care indică rata minimă de eșantionare a unui semnal analogic pentru ca acesta să poată fi reconstituit. Potrivit acestei teoreme, această rată este de două ori mai mare decât cea mai semnificativă frecvență a semnalului.

O variantă de codificare este dată de modularea folosind impulsuri de cod. Tehnica poartă denumirea de *PCM* (*Pulse Control Modulation*).

Algoritmul de codificare constă în doi pași; inițial semnalul analogic este eșantionat cu o rată mai mare sau egală cu dublul frecvenței semnalului; eșantioanele analogice obținute sunt aproximate (cuantificate) folosind una dintre valorile care pot fi reprezentate folosind numărul de biți ales pentru reprezentare.

De exemplu, în cazul unui semnal analogic care are amplitudini cuprinse între 0 și 7 volți, se pot folosi trei biți pentru reprezentare. Astfel, există opt niveluri pentru cuantificare, iar valoarea analogică este aproximată la cel mai apropiat dintre ele.

Calitatea semnalului obținut va fi cu atât mai mare cu cât crește numărul biților folosit pentru reprezentare. Totuși, creșterea numărului de biți duce la creșterea costului transmisiei.

Interconectarea calculatoarelor

O importanță foarte mare în realizarea interconectării calculatoarelor o au canalele de comunicație. Acestea se bazează pe mediile de transmisie și sunt caracterizate cu ajutorul mai multor parametri, cum ar fi tipul legăturii sau sensul de transfer.

În funcție de tipul legăturii realizate între emițător și receptor, distingem trei categorii de canale de comunicație:

- *canale punct-la-punct*: există un singur emițător și un singur receptor;
- *canale punct-multipunct*: există un singur emițător, dar semnalele pot fi primite de mai mulți receptori;
- *canale cu difuzare* (*broadcast*): există mai mulți emițători și mai mulți receptori.

În funcție de sensul de transfer al informației, canalele de comunicație pot fi clasificate astfel:

- *canale simplex*: informația este transmisă într-un singur sens la un moment dat;
- *canale semi-duplex* (*halfduplex - HDX*): datele sunt generate de o singură stație, dar informațiile de control pot fi trimise simultan și de alte stații;
- *canale duplex* (*fullduplex - FDX*): stațiile pot transmite simultan informații.

O caracteristică importantă a canalelor de comunicație este viteza maximă de transfer. Aceasta depinde de frecvența maximă permisă pe acest canal și, în cazul în care canalul nu este afectat de zgomot (cazul ideal), este dată de teorema lui *Nyquist*:

$$v = 2 \cdot H \cdot \log_2 N,$$

unde H este banda de frecvență, iar N este numărul de niveluri folosit pentru codificarea informației.

În cazul în care pe canal pot apărea erori (cazul real) viteza este dată de teorema lui *Shannon*:

$$v = H \cdot \log_2(1 + S / N),$$

unde H este banda de frecvență, S este puterea semnalului util, iar N este puterea semnalului de zgomot.

Unitatea de măsură pentru viteza de transfer este *bitul pe secundă* și este notată prin ***bps***. Raportul semnal/zgomot este dat, de obicei, prin relația $20 \cdot \lg S / N$ și se măsoară în *decibeli* (db).

Multiplexarea datelor

În cazul în care capacitatea de transmisie a mediului de comunicație depășește necesarul de transmisie a unui singur canal, se folosește o tehnică numită multiplexare. Aceasta constă în transmiterea simultană de către mediu a mai multor semnale (corespunzătoare mai multor canale). Există două modalități de realizare a multiplexării:

- multiplexarea cu divizarea frecvenței (*FDM - Frequency-Division Multiplexing*);
- multiplexarea cu divizarea timpului (*TDM - Time-Division Multiplexing*).

Tehnica *FDM* se bazează pe faptul că gama frecvențelor disponibile depășește gama frecvențelor necesare pentru un singur semnal. Fiecare semnal folosește o anumită gamă de frecvențe; frecvențele purtătoarelor sunt suficient de depărtate unele de altele pentru ca frecvențele semnalelor (centrate în jurul frecvenței purtătoarelor) să nu se poată suprapune. Semnalele sursă sunt preluate de un multiplexor, sunt modulate pe o anumită gamă de frecvențe, sunt transmise, sunt preluate de un alt multiplexor și sunt demodulate și refăcute.

Tehnica *TDM* se bazează pe faptul că rata maximă de transmisie este superioară ratei de transmisie de care are nevoie un anumit canal. Din aceste motive, pe canalul de comunicație pot fi "întrețesute" mai multe semnale. Semnalul nu folosește o anumită gamă de frecvențe, ci folosește pentru un timp scurt întreaga bandă. Pentru realizarea unei astfel de transmisii, semnalele sunt împărțite în mai multe componente numite ***felii*** sau ***bucăți***. Termenul folosit în limba engleză, preluat și de limba română este *slot*.

Întreţeserea *slot*-urilor se realizează la nivelul unor blocuri de octeţi. Un ciclu de transmise conţine câte un *slot* de la fiecare sursă şi poartă denumirea de ***cadru*** (engl. ***frame***).

Dacă durata corespunzătoare unui *slot* este fixă, atunci tehnica *TDM* este ***sincronă***; în caz contrar ea este ***asincronă***.

Transmisii sincrone şi asincrone

Pentru ca schimbul de informaţii între emiţător şi receptor să fie corect, este necesar ca receptorul să cunoască exact momentul în care începe primirea unui bit şi durata corespunzătoare acestuia.

O modalitate de transmitere a datelor este ***transmisia asincronă***, care se bazează pe trimiterea datelor sub forma unor caractere. Un ***caracter*** este o secvenţă care conţine între cinci şi opt biţi; dacă se foloseşte acest tip de transmisie, fiecare caracter este precedat de un ***cod de start*** (semnal corespunzător valorii `0` cu o durată echivalentă cu cea a unui bit) şi este urmat de un ***cod de stop*** (semnal corespunzător valorii `1` cu o durată ce depinde de sistem; durata poate fi echivalentă cu durata unui bit, a unui bit şi jumătate sau a doi biţi).

Dacă nu transmite informaţii, emiţătorul generează în continuu coduri de stop. Receptorul identifică începerea transmisiei unui caracter atunci când apare o tranziţie de la valoarea `1` la valoarea `0`. Este foarte important ca receptorul să cunoască durata corespunzătoare unui bit; marja de eroare admisă este de 1%. Acest tip de transmisie se numeşte ***asincronă*** deoarece caracterele pot fi transmise independent unul de altul, cu rate diferite de transmisie.

Pentru a elimina încărcarea suplimentară a liniilor cu secvenţe de start şi stop, a fost introdusă transmisia sincronă. În acest caz nu se lucrează cu caractere, ci cu ***blocuri*** de biţi. Datorită lipsei secvenţelor de start şi stop, este evidentă necesitatea de sincronizare între emiţător şi receptor. Pentru aceasta se foloseşte o linie de transmisie suplimentară pentru semnalul de ceas.

Un nivel suplimentar de sincronizare este necesar datorită faptului că emiţătorul trebuie să determine cu exactitate începutul şi sfârşitul fiecărui bloc. Din acest motiv, fiecare bloc va conţine un câmp de început (numit preambul) a cărui structură este dată de un şablon şi un câmp de sfârşit (numit postambul) cu o structură dată tot de un şablon. Structura care conţine preambulul, informaţiile propriu-zise şi postambulul poartă denumirea de ***cadru***.

Modelul ISO/OSI

În anul 1977, *Organizaţia Internaţională pentru Standardizare* a lansat proiectul ***OSI*** (*Open System Interconnection*). Acesta a devenit un standard care propunea un model de referinţă pentru interconectarea sistemelor deschise, documentul principal fiind *ISO* 7498.

Arhitectura de referinţă este concepută în aşa fel încât să permită conectarea oricăror două sisteme prin intermediul unor interfeţe referinţă. Prin intermediul acestora sunt realizate legăturile între sisteme. Comportarea sistemului se bazează pe un model de referinţă, un model abstract folosit pentru construirea sistemelor capabile de a comunica între ele. Modelul de referinţă *ISO/OSI* este unul abstract, orientat pe conexiune, sistemele fiind considerate baza comunicaţiilor.

Elementele modelului

Aşa cum am spus, baza comunicaţiilor este ***sistemul***. Acesta constă din unul sau mai multe calculatoare, împreună cu *software*-ul aferent, terminalele, perifericele şi aplicaţiile aflate în execuţie.

Un al doilea element important este ***entitatea aplicaţie***. Aceasta este o componentă a unei aplicaţii aferentă unuia dintre nivelurile arhitecturale ale sistemului. Se poate defini cooperarea dintre entităţi din cadrul aceluiaşi nivel, din cadrul a două niveluri adiacente sau din cadrul unor niveluri pereche (acelaşi nivel în două sisteme diferite). O entitate poate fi definită ca fiind un obiect capabil de a schimba informaţii.

Un rol foarte important în cadrul acestui model îl are ***conexiunea***, folosită pentru a realiza legăturile între entităţile cooperante.

O ultimă componentă este ***mediul de interconectare*** care reprezintă modelul mediului fizic folosit pentru realizarea comunicării.

Funcţii

Pentru realizarea unor sisteme distribuite (cum este cazul reţelelor de calculatoare) trebuie definite diferite funcţii complexe. Modelul *ISO/OSI* priveşte aceste funcţii ca fiind o mulţime structurată, folosindu-se în acest scop ***tehnica stratificării***. Un sistem este format din subsisteme care sunt grupate pe ***niveluri ierarhice*** (***straturi***). Specificarea interconectărilor implică folosirea a două tipuri de relaţii. Prima categorie constă în relaţiile definite între nivelurile pereche; ele sunt realizate "*pe orizontală*". A doua categorie constă în relaţiile definite între nivelurile adiacente; acestea sunt realizate "*pe verticală*".

Acest model propune o arhitectură formată din şapte niveluri. Puteţi vedea în figura următoare o reprezentare schematică a acesteia.

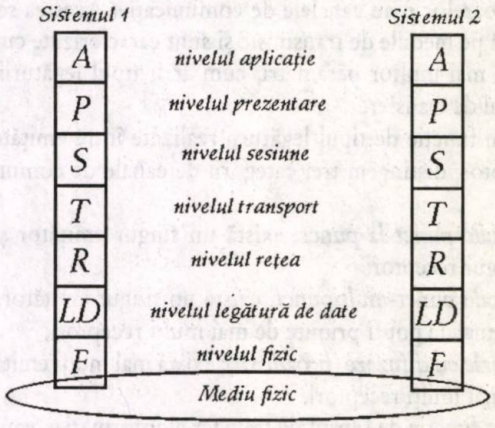


Fig. 8: Modelul arhitectural ISO/OSI

Nivelul aplicație

Acest nivel conține programele care permit utilizatorului să folosească rețeaua. El cuprinde o serie de protocoale standard folosite pentru a rezolva problemele de incompatibilitate apărute datorită unor implementări diferite ale aceluiași tip de aplicație.

Nivelul prezentare

Acest nivel oferă servicii de transformare asupra datelor. Se operează la nivel sintactic și semantic asupra informațiilor transmise. La acest nivel se oferă servicii de compresie și codificare a datelor.

Nivelul sesiune

Acest nivel se ocupă cu organizarea dialogului între aplicațiile care comunică în rețea. Sunt oferite funcții pentru transferul datelor, controlul dialogului, sincronizare etc.

Nivelul transport

Acest nivel realizează legătura între nivelurile superioare (care sunt independente de modul în care se realizează comunicația) și nivelurile superioare (care depind de tehnologia folosită pentru realizarea comunicației). La acest nivel se realizează transferul informațiilor schimbate la nivelul sesiune; principalele funcții ale acestui nivel sunt fragmentarea datelor, transferul lor la nivelul inferior și realizarea controlului erorilor și al fluxului de date.

Nivelul rețea

Acest nivel realizează transferul datelor printr-o rețea de comunicație. Sunt formate pachetele de date care sunt dirijate în rețea; la acest nivel poate fi realizată și interconectarea rețelelor.

Nivelul legătură de date

Acest nivel furnizează nivelului superior o linie de date fără erori de transmisie. Sunt formate unități de date numite cadre a căror trimitere este gestionată prin controlul fluxului și al erorilor.

Nivelul fizic

Acest nivel oferă interfața fizică între echipamente și specifică regulile de transmitere a biților prin canalele de comunicație. La acest nivel sunt prezente interfețele mecanice, electrice, procedurale și funcționale care sunt folosite pentru realizarea transferului informațiilor folosind mediul de transmisie existent.

Structura nivelurilor

Nivelurile arhitecturale ale modelului de referință *ISO/OSI* sunt compuse din trei categorii de elemente:

- ***entitățile*** sunt unitățile funcționale ale nivelului care participă la realizarea funcțiilor acestuia;
- ***protocoalele*** sunt folosite pentru a controla cooperarea entităților;
- serviciile descriu funcțiile nivelului.

Relațiile între niveluri sunt descrise sub forma unei cereri a unui strat inferior (***utilizator de servicii***) către stratul superior adiacent (***furnizor de servicii***). Interacțiunea dintre cele două niveluri este realizată prin intermediul unei primitive abstracte de serviciu (*ASP - Abstract Service Primitive*). O astfel de primitivă constă, din punct de vedere sintactic, din inițiala stratului *OSI* la care se face referire, denumirea colectivă a grupului de primitive din care face parte și numele primitivei. Pentru a nu se crea confuzii, cele trei componente pot fi separate prin delimitatori (`'-'` sau `'.'`). Un exemplu de primitivă este: `T-Connect.req`. Aceasta este o cerere de stabilire a conexiunii la nivelul transport.

Utilizatorii de servicii emit întotdeauna primitive de tipul *request* sau *response*, în timp ce furnizorii emit primitive de tipul *indication* sau *confirmation*.

Interfețele dintre două niveluri adiacente folosesc pentru realizarea transferului de informații așa-numitele ***puncte de acces la servicii*** (*SAP - Service Access Points*).

Specificarea serviciilor unui nivel este dată de descrierea primitivelor aferente stratului. Această specificare este una conceptuală, ea descriind doar efectul primitivei care trebuie să fie independent de interfețele cu straturile adiacente.

Transferul datelor este realizat prin intermediul ***unității de date a protocolului*** (*PDU - Protocol Data Unit*). Această unitate este gestionată în cadrul nivelului și este transmisă nivelului pereche. Practic, această transmisie pe orizontală este realizată printr-o serie de transmisii pe verticală. Fiecare nivel al emițătorului va adăuga la această unitate informații de control proprii. Datele ajung la nivelul fizic de unde sunt transmise către receptor sub forma unui flux de biți. În cadrul sistemului receptor, informațiile parcurg traseul de la nivelul fizic la nivelul aplicație. Fiecare nivel elimină informația de control inserată de nivelul pereche și trimite structura de date obținută către nivelul adiacent superior.

Protocoale

Prin protocolul aferent unui strat se înțelege setul de reguli sintactice și semantice prin intermediul cărora se realizează schimbul de informații între entitățile stratului, astfel încât funcțiile aferente nivelului respectiv să poată fi realizate.

Regulile sintactice descriu formatele mesajelor prin intermediul cărora comunică entitățile. Regulile semantice prezintă conținutul care descrie funcționalitatea stratului respectiv.

Fiecare protocol este caracterizat cu ajutorul unor parametri *cantitativi* și *calitativi*.

Parametrii cantitativi se referă la lungimea unităților de date, dimensiunile mesajelor, cadrelor sau pachetelor, duratele fazelor de conectare, transfer de date, deconectare etc.

Parametrii calitativi se referă la comportarea corectă (lipsa erorilor logice sau funcționale) a protocolului.

Descrierea unui protocol implică prezentarea serviciilor furnizate stratului adiacent superior și a celor solicitate stratului adiacent inferior.

Alte modele de referință

Înaintea elaborării modelului *OSI* existau mai multe modele arhitecturale care erau foarte diferite unele de altele. Datorită diferențelor esențiale era greu de realizat o conectare a unor echipamente care făceau parte din familii diferite.

Cele mai răspândite modele sunt *DNA*, *SNA* și *DPA*. Acesta din urmă este cunoscut și sub denumirea *TCP/IP*, denumire care provine de la numele straturilor aflate la mijlocul ierarhiei de niveluri.

În imaginea următoare puteți vedea o prezentare comparativă a celor patru modele: *OSI*, *DNA*, *SNA* și *DPA*.

OSI	DPA	SNA	DNA
aplicație	procese aplicație	funcții gestionare servicii	utilizator aplicații de rețea
prezentare			
sesiune		control flux date	servicii rețea
transport	TCP, UDP	control transmisie	
rețea	Internet IP+ICMP	control cale	transport dirijare
legătură de date	Acces la rețea LLC	legătură date SDLC	legătură date DDCMP
fizic	fizic	fizic	fizic

Fig. 9: Modelele arhitecturale OSI, DPA, SNA și DNA

Modelul IEEE 802

În ultimii ani s-a observat o tendință de folosire a recomandărilor *OSI* pentru nivelurile superioare ale arhitecturilor. Diferențele apar, în special, la nivelul *fizic* și la nivelul *legăturii de date*. O posibilă ierarhie generală a unui model pentru rețele pune în evidență existența unui nivel fizic care cuprinde aspectele referitoare la comunicația pe un mediu fizic existent și a două subniveluri corespunzătoare nivelului legăturii de date. Primul dintre acestea este ***controlul accesului la mediu*** (*MAC - Medium Acces Control*) și depinde de modul de acces la mediul de transmisie. Cel de-al doilea este ***controlul legăturii logice*** (*LLC - Logical Level Control*) și realizează interfața cu nivelurile superioare.

Prezentare generală

Pentru proiectul *IEEE* 802 au existat șase comitete de standardizare:

- 802.1: probleme legate de interfața cu nivelurile superioare și cu gestionarea rețelei locale;
- 802.2: definirea controlului legăturii logice;
- 802.3: elaborarea standardului *CSMA/CD* (*Carrier Sense, Multiple Access, Collision Detection*);
- 802.4: elaborarea standardului pentru rețelele *Token Bus*;
- 802.5: elaborarea standardului pentru rețelele *Token Ring*;
- 802.6: elaborarea standardului pentru rețelele metropolitane de tip *DQDB* (*Distributed Queue, Dual Bus*).

Ulterior, au apărut alte opt comitete:

- 802.7: avizarea problemelor rețelelor de bandă largă;
- 802.8: avizarea problemelor rețelelor optice;
- 802.9: grupul pentru rețele cu servicii integrate (cu integrarea datelor și a vocii);
- 802.10: securitatea rețelelor;
- 802.11: grupul pentru rețele nebazate pe fir;
- 802.12: grupul pentru rețele de mare viteză de tip *100baseVG*;
- 802.13: grupul pentru rețele de mare viteză de tip 100baseX;
- 802.14: standardizarea transportului datelor în rețele bazate pe cablu TV tradițional.

IEEE 802.1

Acest comitet s-a ocupat cu standardizarea specificațiilor generale ale proiectului *IEEE* 802. Se introduce ideea ca rețelele să furnizeze o interfață unificată către nivelul rețea. Pentru aceasta, nivelul legăturii de date este împărțit în cele două subniveluri amintite anterior: subnivelul controlului legăturii logice și subnivelul controlului accesului la mediu. Primul dintre acestea realizează interfața către nivelul arhitectural superior și este descris în standardul *IEEE* 802.2.

IEEE 802.2

Acest comitet a stabilit standardul pentru subnivelul controlului legăturii logice (*LLC*), descriind atât serviciile furnizate de acest nivel, cât și protocolul care le implementează. Există trei tipuri de servicii:

- **LLC Type 1**: nu este bazat pe conexiune, nu folosește achitarea și este foarte rapid;
- **LLC Type 2**: este bazat pe conexiune și folosește achitarea;
- **LLC Type 3**: nu este bazat pe conexiune, dar folosește achitarea.

O rețea de calculatoare poate oferi unul sau mai multe servicii. O posibilă clasificare folosind acest criteriu ar putea fi:

- **rețea de clasa I**: servicii LLC Type 1;
- **rețea de clasa a II-a**: servicii LLC Type 1 și LLC Type 2;
- **rețea de clasa a III-a**: servicii LLC Type 1 și LLC Type 3;
- **rețea de clasa a IV-a**: servicii LLC Type 1, LLC Type 2 și LLC Type 3.

Unitatea de date *LLC-PDU* cuprinde patru câmpuri: ***Adresa Destinație*** (un octet), ***Adresa Sursă*** (un octet), ***Control*** (unul sau doi octeți) și ***Informație*** (număr variabil de octeți).

Câmpul *Control* identifică trei tipuri de unități de date:

- *I-PDU*: unități care conțin date propriu-zise;
- *S-PDU*: unități care conțin date de control;
- *U-PDU*: unități care conțin date necesare pentru inițializări sau diagnosticări.

Deoarece există o gamă diversă de protocoale care pot fi folosite la nivelurile superioare, protocolul *LLC* folosește un ***punct de acces la servicii*** (*SAP - Service Access Point*) numit *LLC-SPA*. Acesta apare în câmpul de adrese al unității de date *LLC-PDU* și este folosit pentru adresarea folosită de *LLC* pentru a face diferența între protocoalele de nivel superior utilizate.

Controlul accesului la mediu

Subnivelul *MAC* este specific fiecărui tip de rețea și are funcția de gestionare a partajării mediului de transmisie între sistemele care formează rețeaua. Există mai multe metode de acces la mediu, bazate pe diferite principii.

Fiecare tip de rețea are formate proprii pentru unitățile de date ale protocolului *MAC*. Totuși, există anumite câmpuri comune; oricare *MAC-PDU* trebuie să conțină câmpurile din ***DSAP***, ***SSAP***, ***Info*** și ***FCS***.

Câmpurile *DSAP* și SSAP *indică* adresa destinație și adresa sursă a transferului. Câmpul *Info* conține datele propriu-zise care trebuie transferate. Aceste date sunt cele primite de la subnivelul de control al legăturii logice, deci au formatul *LLC-PDU*. Câmpul *FCS* (*Frame Control Sequence*) conține 32 de biți și este folosit pentru controlul erorilor de transmisie.

Secvența de control este dată de polinomul generator aferent codului ciclic redundant (***CRC*** - *Cyclic Redundancy Check*). Se pornește cu un polinom ai cărui coeficienți sunt dați de valorile biților corespunzători câmpurilor verificate. Secvența *FCS* este dată de restul împărțirii acestui polinom la un polinom standard. Standardul *IEEE* 802 folosește următorul polinom:

$$CRC_{32} = X^{32} + X^{26} + X^{25} + X^{22} + X^{16} + X^{12} + X^{11} + \\ + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1.$$

Adresarea MAC

Adresele *MAC* au o lungime de șase octeți, sunt înscrise în memoria *ROM* a plăcii de rețea și sunt stabilite la nivel mondial de către *IEEE*. Ulterior, ele pot fi modificate prin *software*, folosindu-se un *buffer* care conține noua adresă.

Primii trei octeți identifică producătorul plăcii de rețea sau organizația care a proiectat acel tip de rețea. Acest câmp poartă denumirea ***OUI*** (*Organization Unique Identifier*). Ceilalți trei octeți sunt completați de producător și reprezintă un număr de ordine.

Primii doi biți transmiși (*I/G* și *U/L*) au o importanță specială.

Bitul *I/G* (*Individual/Grup*) arată tipul destinației: o singură stație (valoarea `0` - *Individual*) sau un grup de stații (valoarea `1` - *Grup*).

Bitul *U/L* (*Universal/Local*) arată dacă adresa a fost atribuită oficial de *IEEE* (valoarea `0` - *Universal*) sau a fost atribuită local (valoarea `1` - *Local*).

Rețele locale de calculatoare

O posibilă definiție a rețelelor locale de calculatoare (*LAN - Local Area Networks*) ar putea fi următoarea:

Definiție

O rețea locală de calculatoare este un sistem care permite unor aparate independente să comunice între ele, în cadrul unui spațiu limitat, utilizând pentru comunicație canale fizice sigure, de mare viteză.

Aparate independente

Această noțiune indică orice aparat (sistem) care este conectat la rețea și schimbă date cu celelalte sisteme aflate în aceeași rețea.

Zonă limitată

Acest termen indică faptul că elementele componente ale rețelei se află la o distanță relativ mică unele de altele.

Canale sigure, de mare viteză

Dorința utilizatorilor de a avea condiții de lucru optime, implică folosirea unor medii de transmisie cu viteze cât mai mari. Acestea trebuie să aibă o calitate ridicată, ceea ce duce la o rată mică a erorilor.

Caracteristicile LAN-urilor

Una dintre cele mai importante caracteristici a unei astfel de rețele este ***flexibilitatea***. Aceasta permite utilizarea rețelei în aplicații diverse deoarece ea poate conține echipamente diverse. Un rol important îl are ***siguranța***, care este dată de o tehnologie bine pusă la punct. ***Modularitatea*** oferă posibilitatea integrării în rețea a unor echipamente diverse; acestea sunt conectate prin intermediul unor plăci de interfațare (***NIC*** - *Network Interface Card*). ***Expandabilitatea*** unei rețele se referă la faptul că se poate crește complexitatea acesteia, în funcție de necesitățile utilizatorilor. Este preferabilă interconectarea mai multor rețele în locul dezvoltării unei singure rețele cu o complexitate ridicată. Administrarea și gestionarea unei rețele locale este relativ simplă.

Topologiile rețelelor

Noțiunea de topologie a unei rețele se referă la modul în care sunt realizate legăturile de comunicație între stațiile care formează rețeaua. Cele mai cunoscute topologii folosite sunt prezentate în figura alăturată.

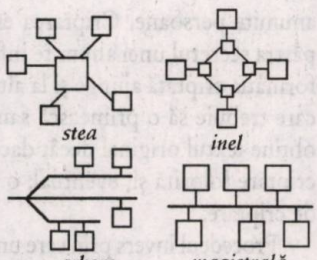


Fig. 10: Topologii de rețele locale

În numărul următor...

Luna viitoare vom continua prezentarea rețelelor de calculatoare cu descrierea mediilor de transmisie folosite și a modalităților de cablare a unei rețele.

Mihai Scorțaru este redactor-șef adjunct al GInfo. Poate fi contactat prin e-mail la adresa **skortzy@yahoo.com**.

Tehnici de CRIPTARE

Claudiu Soroiu

Criptarea mesajelor este unul dintre cele mai studiate domenii din zilele noastre. Complexitatea tehnicilor de criptare și de trimitere a mesajelor crește permanent pentru a obține o securitate cât mai ridicată a comunicațiilor. Pe parcursul câtorva episoade, vom prezenta unele dintre cele mai cunoscute modalități de codificare a mesajelor.

Introducere

Criptarea mesajelor și trimiterea lor sub această formă este utilizată de foarte mult timp. Unul dintre primii care au folosit tehnici de criptare pentru trimiterea mesajelor a fost celebrul împărat roman *Cezar*. În comunicarea privată cu o persoană, folosirea unui cod secret poate preveni citirea intenționată sau neintenționată a mesajelor de către cei care intră în posesia acestora, fie pentru că trebuie să le transporte până la destinatar, fie pentru că le interceptează în timp ce mesajul este transmis.

Definiție

Criptarea reprezintă o metodă prin care un *text necriptat* (documente, fișiere, fraze, cuvinte etc.) este transformat, utilizând anumite metode, într-un alt text, numit ***text criptat***, care se păstrează în această formă sau se trimite unei anumite persoane. Criptarea este utilizată cu scopul de a păstra secretul unei anumite informații. În cazul în care informația criptată ajunge și la alte persoane, altele decât cele care trebuie să o primească sau să o păstreze, nu se poate obține textul original decât dacă este cunoscută tehnica de criptare folosită și, eventual, o parolă utilizată în procesul de criptare.

Procedeul invers prin care un text criptat este transformat în textul original poartă numele de ***decriptare***.

Există foarte multe tehnici de criptare cunoscute de foarte multe persoane, dar cu toate acestea, pentru tehnicile avansate, datorită complexității algoritmilor folosiți pentru codificare, procesul invers, cel de *decriptare*, este aproape imposibil de realizat.

Un algoritm de criptare este cu atât mai eficient, cu cât textul criptat și eventuala parolă utilizată sunt foarte greu de găsit, chiar dacă cel care încearcă să decripteze textul deține o parte din textul inițial. Un alt factor important de care depinde eficiența unui algoritm de criptare implementat într-un anumit limbaj de programare este timpul folosit de programul respectiv pentru a codifica o cantitate foarte mare de informații. Un text criptat folosind o metodă care nu necesită cunoștințe vaste din domeniul matematicii și informaticii este foarte ușor de decriptat.

Criptarea lui Cezar

Când trimitea mesaje oficiale către subalternii săi, *Cezar* folosea o tehnică foarte simplă de codificare a acestora. Fie două cercuri pe care sunt scrise cele 26 de litere ale alfabetului englezesc; dacă rotim cercul mic cu un număr de litere față de celălalt în sensul acelor de ceasornic, atunci, pentru a cripta un mesaj, se caută fiecare literă din mesaj pe cercul mare și se scrie litera corespunzătoare de pe cercul mic.

Figura 1

Figura 2

Dacă avem textul *IULIUS CAESAR* și deplasăm cercul din interior din figura 1 cu două litere în sensul acelor de ceasornic după cum se observă în figura 2, atunci textul criptat va fi *GSJGSQ AYCQYP*.

Procedeul de rotire a cercului mic cu un anumit număr de litere poartă numele de ***deplasare***. Pentru această tehnică de criptare, o parte din cheia (***parola***) de criptare o constituie numărul de deplasări (***deplasament***) de câte o literă în sensul acelor de ceasornic. Cealaltă parte din cheia de

criptare o constituie alfabetul ales. Alfabetul folosit poate să fie altul decât cel prezentat în exemplul anterior. Acesta poate să conțină pe lângă majuscule și alte caractere printre care pot apărea cifre, caractere speciale, alte litere ale alfabetelor diferitelor limbaje utilizate sau poate să fie întreg setul de caractere *ASCII* folosit de calculator.

Pentru a decripta un mesaj criptat cu această metodă, trebuie să se cunoască numărul de deplasări utilizat la criptarea textului inițial. Procesul de obținere a mesajului inițial din mesajul criptat când se cunoaște deplasamentul este foarte simplu. Se deplasează cercul din interior cu atâtea litere cu câte indică deplasamentul și fiecare literă din textul criptat se caută pe cercul din interior și se scrie litera corespunzătoare de pe cercul din exterior. Din textul *FRIFRP ZXBPXO* criptat în cazul alfabetului englezesc, folosind un deplasament de trei caracactere, prin decriptare se obține *IULIUS CAESAR*.

Dacă notăm cu L numărul de caractere al alfabetului utilizat, atunci textul criptat folosind un deplasament D mai mare sau egal cu L este același cu cazul în care deplasamentul este restul împărțirii lui D la L.

Această metodă de criptare nu este eficientă, deoarece textul criptat poate fi ușor decriptat dacă se cunoaște alfabetul utilizat. La decriptare se folosesc L deplasamente și astfel se generează L texte dintre care numai unul este cel bun și poate fi ușor identificat.

Variante ale criptării lui Cezar

O variantă a criptării lui *Cezar* este tehnica folosită de *Ovidius*. Acesta nu folosea deplasarea cercului din interior, în schimb literele de pe cercul din interior erau scrise în ordine inversă (de la ultima literă din alfabet la prima, în sensul acelor de ceasornic). Acest lucru este echivalent cu scrierea literelor din alfabet pe un rând în ordine și pe celălalt rând în ordine inversă, după cum se poate observa în continuare:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Z Y X W V U T S R Q P O N M L K J I H G F E D C B A

Pentru a cripta cuvântul *OVIDIUS* se caută pentru fiecare literă din cuvânt litera de pe primul rând și se scrie litera de pe al doilea rând de sub litera găsită. Aplicând acest procedeu obținem cuvântul *MERWRFH*.

Pentru decriptare se folosește același procedeu ca în cazul criptării deoarece, dacă unei litere i de pe primul rând îi corespunde a $(L + 1 - i)$-a literă din al doilea rând, atunci celei de-a $(L + 1 - i)$-a litere din primul rând îi corespunde a $(L + 1 - (L + 1 - i))$-a literă din al doilea rând, adică a i-a literă.

Criptarea cu alfabet aleator

O altă variantă a criptării lui *Cezar* este folosirea unui alfabet și aranjarea caracterelor acestuia, pe cele două cercuri, într-o ordine aleatoare, dar caracterele de pe ambele cercuri trebuie să fie aranjate în aceeași ordine.

Pentru decriptarea mesajelor criptate cu ajutorul unui alfabet ales aleator, trebuie cunoscute alfabetul, ordinea caracterelor din alfabet precum și numărul de deplasări. Având aceste componente, procesul de decriptare este similar celui prezentat în secțiunea anterioară.

Dacă se folosește alfabetul englezesc, ca posibilă modalitate de aranjare a caracterelor pe cele două cercuri ar putea fi aleasă aranjarea lor de pe tastatură, și anume:

Q W E R T Y U I O P A S D F G H J K L Z X C V B N M

În cazul în care se cunoaște numai tehnica de criptare utilizată, decriptarea mesajului este mai complicată, deoarece se bazează pe statistică matematică și în principal pe faptul că într-un text necriptat cu o dimensiune suficient de mare vocalele au o frecvență foarte mare și nu există două vocale consecutive din text care să fie la o distanță mai mare de șase caractere (pentru limba română).

Metoda substituției

Prin *substituție* se înțelege înlocuirea fiecărui caracter din alfabet cu un altul și nu există două caractere care să fie înlocuite cu același caracter. O altă posibilitate ar fi aceea de a înlocui fiecare caracter din alfabetul ales cu un simbol sau un număr.

Tehnicile de criptare prezentate până acum sunt cazuri particulare ale criptării prin substituție.

Considerând alfabetul englezesc, există 26! posibilități de substituire a caracterelor cu altele din care, pentru o mai mare securitate a datelor se scade numărul de posibilități în care un caracter este substituit cu el însuși (acest număr se obține rezolvând *problema concordanțelor*).

Criptarea unui text folosind această metodă se face ca în cazul criptării lui *Ovidius*, adică pentru fiecare caracter din text se scrie litera care s-a ales pentru substituție.

Fie următoarea aranjare a caracterelor alfabetului englezesc:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Z I R A J S B K T C L U D M V E N W F O X G P Y H Q

Pentru textul *PAROLA ESTE ASDFG*, mesajul criptat folosind aranjarea de mai sus, este: *NZFEDZ JOXJ ZOASB*.

Procesul de decriptare se face în sens invers, adică pentru fiecare literă din textul criptat se scrie litera pe care aceasta a substituit-o în procesul de criptare.

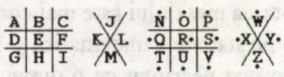
Caracterele se pot substitui și cu numere, existând mai multe variante a acestui procedeu. De exemplu, fiecărui caracter i se poate asocia o mulțime de numere care îl pot substitui, dar mulțimile asociate caracterelor alfabetului ales trebuie să fie disjuncte. Dacă pentru litera **A** se alege o mulțime $\{n_1, n_2, ..., n_m\}$, oricare din elementele mulțimii pot substitui această literă. În procesul de criptare, pentru fiecare literă din textul inițial se alege aleator un număr din mulțimea care i-a fost atribuită.

De exemplu, dacă pentru caracterul **A** se alege mulțimea {1, ..., 10}, pentru litera **B** se alege mulțimea {11, ..., 20} etc., există mai multe posibilități pentru a cripta textul de mai sus; una dintre ele poate fi: 154, 7, 172, 149, 120, 156, 48, 185, 194, 44, 4, 184, 33, 51, 63.

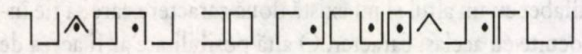
Nu este obligatoriu ca mulțimile alese pentru fiecare caracter să aibă același cardinal; de fapt, este indicat ca

mulțimile asociate literelor care apar frecvent să aibă mai multe elemente.

Pentru transmiterea de texte criptate se mai pot folosi imagini în care caracterele din textul inițial sunt înlocuite cu simboluri predefinite. Una dintre cele mai cunoscute metode de înlocuire a caracterelor cu simboluri este prezentată în figura următoare.



Pentru fiecare caracter din figură se desenează conturul regiunii în care acesta se află. De exemplu, pentru litera N se desenează laturile regiunii din stânga-sus ale diagramei în care se află împreună cu un punct între laturi. Folosind aceste simboluri, textul *GAZETA DE INFORMATICA* se scrie astfel:



Metoda transpoziției

Criptarea prin *metoda transpoziției* este o tehnică mai eficientă decât criptarea prin substituție, dar are, la rândul ei, o mulțime de dezavantaje.

Textul criptat prin metoda transpoziției păstrează toate caracterele textului inițial, dar în altă ordine obținută prin aplicarea algoritmului ce va fi prezentat în continuare.

Criptarea prin transpoziție constă din scrierea textului inițial din care s-au eliminat spațiile și semnele de punctuație, într-o matrice de dimensiune $M \times N$, interschimbarea anumitor linii (sau coloane) între ele și textul criptat se obține prin scrierea caracterelor din noua matrice de pe fiecare coloană în parte, începând cu colțul din stânga-sus. Dacă lungimea textului inițial este mai mică decât numărul de elemente ce pot fi scrise în matrice, atunci textul se completează cu elemente aleatoare, până ajunge la dimensiunea $M*N$.

Pentru textul *Misiunea a fost îndeplinită*, care are lungimea de 24 de caractere, se pot alege mai multe matrice de dimensiune $M \times N$, o posibilitate ar fi ca matricea să aibă 4 linii și 6 coloane, dar pentru ca textul să fie mai greu de decodificat trebuie să conțină și caractere alese aleator, sau într-un mod mai inteligent, care să îngreuneze munca celui care dorește să afle conținutul secret din mesajul criptat. Fie o matrice care are 5 linii și 6 coloane. Textului inițial i se adaugă 6 caractere aleatoare și se obține textul *Misiu neaaf ostîn depli nităx yztwu* și se scrie în matricea din partea stângă astfel:

	1	2	3	4	5	6
1	M	i	s	i	u	n
2	e	a	a	f	o	s
3	t	î	n	d	e	p
4	l	i	n	i	t	ă
5	x	y	z	t	w	u

	1	2	3	4	5	6
5	x	y	z	t	w	u
3	t	î	n	d	e	p
4	l	i	n	i	t	ă
1	M	i	s	i	u	n
2	e	a	a	f	o	s

Prin scrierea liniilor 1, 2, 3, 4, 5 în ordinea 5, 3, 4, 1, 2, se obține matricea din partea dreaptă. Textul criptat care se obține este: *xtlMe yîiia znnsa tdiif wetuo upăns.*

Transpoziție cu parolă

Pentru ca procesul de decriptare să fie mai simplu și să nu mai fie nevoie ca ordinea în care au fost puse liniile din matricea creată, se folosește o variantă a criptării prin transpoziție care se bazează pe o parolă.

Pentru a cripta un text folosind o parolă și metoda transpoziției, se alege o parolă ale cărei litere determină ordinea în care se vor scrie coloanele din matricea aleasă.

Pentru a afla ordinea în care vor fi scrise coloanele din textul inițial, se ordonează alfabetic literele din parolă, și fiecărei litere i se asociază numărul de ordine din șirul ordonat.

Lungimea parolei trebuie să fie egală cu numărul de coloane din matrice.

Considerăm textul anterior, scris într-o matrice de dimensiuni 5 × 6, și parola *vultur*. Literele din parolă se ordonează alfabetic și se obține șirul: *l*, *r*, *t*, *u*, *u*, *v*. Indicele 1 este asociat cu litera *l*, indicele 2 cu litera *r*, indicele 3 cu litera *t*, indicele 4 cu prima literă *u* din parolă, indicele 5 cu a doua literă *u* din parolă, iar indicele 6 este asociat cu litera *v*. Pentru a scrie coloanele, pentru fiecare indice *i* din șirul ordonat se caută indicele *j*, care reprezintă poziția literei cu indicele *i*, din parolă și se scrie coloana *j*, astfel:

	v	u	l	t	u	r
	6	4	1	3	5	2
1	M	i	s	i	u	n
2	e	a	a	f	o	s
3	t	î	n	d	e	p
4	l	i	n	i	t	ă
5	x	y	z	t	w	u

	1	2	3	4	5	6
5	s	n	i	i	u	M
3	a	s	f	a	o	e
4	n	p	d	î	e	t
1	n	ă	i	i	t	l
2	z	u	t	y	w	x

Textul care se obține în final este: *sannz nspău ifdit iaîiy uoetw Metlx.*

Pentru a decripta un mesaj criptat folosind această metodă, mesajul se scrie în matrice pe coloane, începând cu colțul stânga-sus, și apoi se realizează operația inversă, adică pentru fiecare indice *j* al literelor din parolă, se caută indicele *i* asociat literei din șirul sortat și se scrie coloana cu indicele *i*. Din noua matrice astfel obținută se scriu literele de pe fiecare linie, în ordine.

O tehnică foarte cunoscută și foarte practică de transmitere a mesajelor folosind metoda transpoziției constă în înfășurarea unei panglici în jurul unui băț. Mesajul se scrie pe panglică, de-a lungul bățului, de la capătul superior spre capătul inferior, pe coloane și apoi se trimite la destinație numai panglica, care ulterior s-a desfăcut de pe băț. La destinație se înfășoară panglica pe un băț având aceeași dimensiune cu cel care a ajutat la scrierea textului și se citește textul pe coloane.

Această tehnică de transmitere a mesajelor criptate era folosită chiar și în antichitate.

De atunci anii au trecut și tehnicile de criptare au evoluat foarte mult. În numărul următor vom prezenta câteva dintre tehnicile avansate de criptare avansate.

Claudiu Soroiu este redactor al GInfo. Poate fi contactat prin e-mail la adresa csoroiu@yahoo.com.

Introducere în PROCESAREA imaginilor

Radu-Daniel Vatavu

Procesarea imaginilor este un domeniu al inteligenței artificiale care se ocupă cu reprezentarea, reconstituirea, clasificarea, recunoașterea și analiza imaginilor cu ajutorul calculatorului.

Introducere

Sursa de proveniență a imaginilor poate fi un dispozitiv de achiziție (cameră video, scanner, captor radar), dar poate fi la fel de bine și o ecuație matematică, un ansamblu de date statistice etc.

Există două motive principale pentru care se apelează la procesarea imaginilor:

- îmbunătățirea calității unei imagini având drept scop o mai bună vizualizare pentru un operator uman. Aceasta poate însemna: reducerea zgomotului și a altor defecte care pot fi prezente în imagine (datorate, de exemplu, dispozitivului de achiziție), evidențierea unor zone de interes prin modificarea luminozității, a contrastului, accentuarea muchiilor etc.;
- extragerea de informații dintr-o imagine, informații care pot reprezenta intrarea pentru un sistem automat de recunoaștere și clasificare. Aceste informații pot fi: diferite distanțe și relații dintre obiectele prezente în imagine, momente statistice, parametri geometrici (arie, perimetru, circularitate), coeficienți *Fourier* etc.

Exemple clasice de aplicații pentru procesarea imaginilor (și recunoașterea formelor) includ: recunoașterea caracterelor, recunoașterea amprentelor, prelucrarea imaginilor medicale, a imaginilor satelit.

Noțiuni de bază

În cele ce urmează se va considera drept reprezentare a unei imagini un tablou bidimensional de numere întregi (*pixeli*). Valoarea fiecărui pixel descrie nivelul de strălucire sau culoarea acestuia. În cel mai simplu caz, al imaginilor binare, pentru reprezentarea fiecărui pixel este folosit un singur bit. În cazul imaginilor cu niveluri de gri, valoarea fiecărui pixel reprezintă strălucirea acestuia.

Cel mai comun format pentru aceste imagini are la bază reprezentarea pixelilor pe 8 biți. Astfel, gama de valori posibile este 0..255, 0 codificând culoarea ***negru***, 255 ***alb***, iar valorile intermediare reprezentând nuanțe (niveluri) de gri. În cazul imaginilor color, pot fi folosite diferite sisteme de culori (*RGB*, *HSI*, *CMY* etc.).

Modele de culori

Modelul *RGB* este un model aditiv, având culorile fundamentale roșu, verde și albastru, celelalte culori constituindu-se din anumite procente ale culorilor de bază. Culoarea alb are prezente toate culorile fundamentale, iar culoarea negru se caracterizează prin absența lor. Modelul *RGB* este adesea reprezentat sub forma unui cub unitate, ca în figura 1:

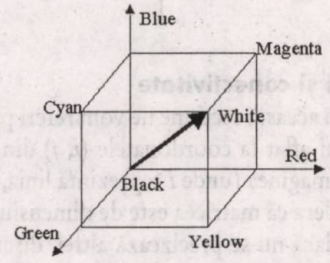


Figura 1

Originea (0, 0, 0) este considerată culoarea ***negru***, iar colțul opus (1, 1, 1) corespunde culorii ***alb***. Diagonala care unește aceste două puncte conține nivelurile de gri obținute prin utilizarea aceluiași procent din culorile de bază.

O altă metodă de clasificare a culorilor este prin ***nuanțare*** (*Hue*), ***saturație*** (*Saturation*) și ***intensitate*** (*Intensity*), rezultând modelul *HSI*.

Nuanța reprezintă poziția relativă a culorii în spectru și, în modelul *HSI*, corespunde unghiului pe care îl face culoarea în cercul de culori. Domeniul de nuanțare se întinde de la 0 la 360 de grade.

Saturația reprezintă puritatea culorii. Valoarea saturației se exprimă în procente și variază de la 0 (fără culoare) până la 100 (culoare pură, așa cum este definită de valoarea *nuanțării*).

Ultimul parametru reprezintă intensitatea culorii. O reprezentare simbolică a modelului *HSI* este dată în figura 2. Nuanța determină culoarea (unghi = 60° - *Yellow*, 180° - *Cyan* etc.) Saturația este măsurată în procente de la centrul cercului de bază la suprafața conurilor.

Intensitatea se măsoară pe linia *Black-White*.

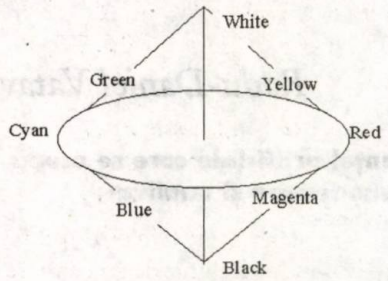


Figura 2

Modelul ***CMY*** are la bază culorile *Cyan*, *Magenta* și *Yellow*, considerate culori secundare față de culorile *Red*, *Green*, *Blue*.

Modelul este substractiv, culoarea *alb* fiind dată de absența culorilor *CMY*, iar culoarea *negru* fiind caracterizată de prezența acestora.

În continuare sunt date câteva modalități de conversie între tipurile de modele prezentate:

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad \begin{bmatrix} H \\ S \\ I \end{bmatrix} = \begin{bmatrix} \dfrac{\frac{\pi}{2} - \arctan\left\{\dfrac{2R-G-B}{\sqrt{3}}(G-B)\right\} + \pi}{2\pi} \\ 1 - \dfrac{3}{(R+G+B)}\min(R,G,B) \\ \dfrac{R+G+B}{3} \end{bmatrix}$$

Vecinătăți și conectivitate

Începând cu această secțiune ne vom referi prin $f_{i,j}$ la valoarea pixelului aflat la coordonatele (i, j) din matricea care reprezintă imaginea (unde i reprezintă linia, iar j coloana). Vom considera că matricea este de dimensiune $M \times N$. De asemenea, dacă nu se precizează altfel, discuțiile vor avea în vedere o imagine cu niveluri de gri (considerăm reprezentarea pe 8 biți, deci cu 256 de valori posibile). Aceasta nu reprezintă o limitare, întrucât o imagine color poate fi descompusă după culorile fundamentale (R, G, B) în trei imagini separate, fiecare astfel de imagine putând fi tratată ca alcătuită din niveluri de gri.

Un pixel p de coordonate (i, j) prezintă patru vecini în punctele: $(i\text{-}1, j)$, $(i\text{+}1, j)$, $(i, j\text{-}1)$, $(i, j\text{+}1)$. Vom nota acest ansamblu (vecinătate) de pixeli cu $N_4(p)$. Pixelii diagonali de coordonate: $(i\text{-}1, j\text{-}1)$, $(i\text{-}1, j\text{+}1)$, $(i\text{+}1, j\text{+}1)$, $(i\text{+}1, j\text{-}1)$ alcătuiesc un nou ansamblu notat $N_D(p)$. În final, ansamblurile $N_4(p)$ și $N_D(p)$ reunite, formează ansamblul notat $N_8(p)$.

Având în vedere ansamblurile (vecinătățile) definite mai sus, se pot considera două tipuri de conectivitate:

- ***4-conectivitate*** - doi pixeli, p_1 și p_2, de coordonate (i_1, j_1) și (i_2, j_2) sunt în relație de *4-conectivitate* dacă (i_1, j_1) se află în ansamblul $N_4(p_2)$ sau echivalent, dacă (i_2, j_2) se află în ansamblul $N_4(p_1)$;
- ***8-conectivitate*** - doi pixeli, p_1 și p_2, de coordonate (i_1, j_1) și (i_2, j_2) sunt în relație de *8-conectivitate* dacă (i_1, j_1) se află în ansamblul $N_8(p_2)$ sau echivalent, dacă (i_2, j_2) se află în ansamblul $N_8(p_1)$.

Se numește drum de lungime n mulțimea de pixeli $D_n=\{p_1, p_2, \ldots p_n\}$ pentru care fiecare pixel p_i se află în relație de *conectivitate* cu pixelul următor p_{i+1}, $i=1 \ldots n\text{-}1$.

Doi pixeli p și q sunt ***conectați*** dacă există un drum D_n de la p la q.

Un set de pixeli formează o ***regiune*** dacă orice doi pixeli aparținând setului sunt *conectați*.

Metrici

Distanțele dintre pixeli poartă denumirea de ***metrici***. Fie P mulțimea pixelilor unei imagini. O *metrică*, în cazul unei imagini, este o funcție $d : P \times P \to R$ care îndeplinește următoarele condiții:

$$d(p_1, p_2) \geq 0 \quad \forall\, p_1, p_2 \in P.\ d(p_1, p_2) = 0 \Leftrightarrow p_1 = p$$
$$d(p_1, p_2) = d(p_2, p_1) \quad \forall\, p_1, p_2 \in P$$
$$d(p_1, p_2) \leq d(p_1, p_3) + d(p_3, p_2) \quad \forall\, p_1, p_2, p_3 \in P$$

Câteva exemple de metrici utilizate frecvent sunt:

- *distanța euclidiană*:

$$d(p_1, p_2) = \sqrt{(i_1 - i_2)^2 + (j_1 - j_2)^2},$$

unde p_1 și p_2 sunt pixeli de coordonate (i_1, j_1), respectiv (i_2, j_2).

- *distanța Manhattan* (*city block metric*):

$$d_4(p_1, p_2) = |i_1 - i_2| + |j_1 - j_2|.$$

În baza acestei distanțe putem da o definiție riguroasă pentru ansamblul $N_4(p)$, astfel:

$$N_4(p) = \{q \in P / d_4(p, q) \leq 1\};$$

- distanța jocului de șah (*chessboard metric*):

$$d_8(p_1, p_2) = \max\{|i_1 - i_2|, |j_1 - j_2|\}$$

Având această funcție, se poate da o nouă definiție pentru ansamblul $N_8(p)$, astfel:

$$N_8(p) = \{q \in P / d_8(p, q) \leq 1\}.$$

Histograma nivelurilor de gri

Histograma nivelurilor de gri a unei imagini reprezintă o funcție care asociază fiecărui nivel de gri prezent în imagine frecvența sa de apariție.

Dacă luăm în considerare o imagine f care prezintă un număr maxim G de niveluri de gri, atunci funcția histogramă poate fi definită astfel:

$$h: \{0,1,...G-1\} \rightarrow \{0,1,...N \times M\}$$

$$h(k) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [f_{i,j} = k], \quad \text{unde } [x = y] = \begin{cases} 1, & \text{pentru } x = y \\ 0, & \text{pentru } x \neq y \end{cases}$$

Un exemplu de histogramă este dat în figura următoare:

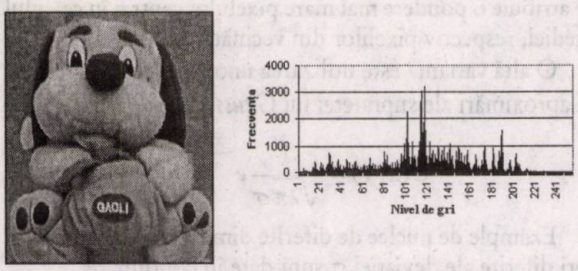


Figura 3: Imagine cu 256 de niveluri de gri și histograma nivelurilor de gri

Histograma nu oferă nici o informație cu privire la locația pixelilor sau la poziția relativă a acestora, însă, dispunând de histograma unei imagini, nivelurile de gri pot fi manipulate pentru a îmbunătăți contrastul, luminozitatea, etc. De exemplu, o deplasare a nivelurilor de gri din histogramă spre stânga/dreapta conduce la o diminuare/creștere a luminozității.

Se poate observa din figura 3 o concentrare a valorii unor pixeli în jurul unor anumite valori (de exemplu 120), reprezentând regiuni cu aceeași luminozitate sau luminozități foarte apropiate (cum ar fi fondul imaginii). Acest aspect este foarte util în procesul de segmentare permițând o separare rapidă a obiectelor de fundal în funcție de luminozitatea acestora.

În cazul imaginilor color, folosind modelul *RGB*, se pot realiza histograme pentru fiecare culoare de bază în parte cu păstrarea validității celor discutate mai sus.

Preprocesarea imaginilor

Preprocesarea are drept scop realizarea unei îmbunătățiri a imaginii prin punerea în evidență a anumitor regiuni, schimbarea luminozității, detecția muchiilor etc., pregătind imaginea în vederea operației de segmentare.

Transformările aplicate în această etapă unei imagini pot fi de două tipuri:

- transformări punctuale (modificarea valorii unui pixel se face independent de vecinătatea acestuia);
- transformări locale (noua valoare a pixelului depinde de valorile pixelilor înconjurători).

Transformări punctuale

Din cadrul transformărilor punctuale vom avea în vedere manipularea contrastului prin transformarea nivelurilor de gri. În acest sens, o transformare punctuală poate fi privită ca o funcție de transfer T definită pe mulțimea nivelurilor de gri originale, care atribuie fiecărui pixel o nouă valoare în funcție de luminozitatea sa. Exemple de funcții de transfer sunt date în următoarea figură:

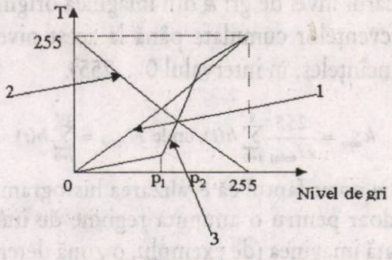


Figura 4

- funcția identitate: $T(p) = p$, $p = 0 \ldots 255$;
- funcția inversă: $T(p) = 255\text{-}p$, $p = 0 \ldots 255$;
- o funcție de tipul:

$$T(p) = \begin{cases} a_1 p + b_1, & \text{pentru } p < p_1 \\ a_2 p + b_2, & \text{pentru } p_1 \leq p \leq p_2 \\ a_3 p + b_3, & \text{pentru } p > p_2 \end{cases}$$

Funcția 3 prezentată poate fi folosită pentru accentuarea contrastului dintre nivelurile de gri cuprinse între p_1 și p_2, dacă panta a_2 este superioară valorilor a_1 și a_3.

Egalizarea histogramei

Transformările nivelurilor de gri au drept scop o mai bună vizualizare a imaginii de către un operator uman. O tehnică din cadrul acestor transformări, care dă rezultate bune în practică, poartă denumirea de *egalizarea histogramei* și constă în distribuirea nivelurilor de gri existente pe întreaga gamă posibilă de valori.

Această tehnică este foarte utilă în cazul unor imagini ale căror niveluri de gri sunt localizate într-un interval restrâns, iar "împrăștierea" lor pe întreaga scală conduce la o vizualizare sensibil mai bună. Funcția de transfer utilizată în acest caz are la bază histograma nivelurilor de gri, fiind reprezentată de curba cumulată a frecvențelor nivelurilor de gri.

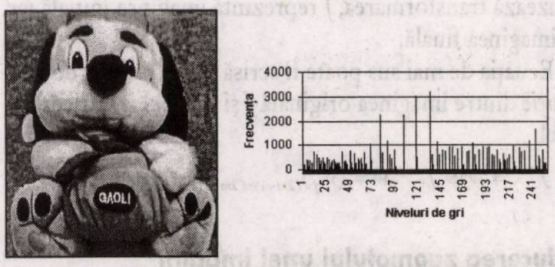


Figura 5: Imaginea după egalizarea histogramei și histograma după egalizare

În figura 5 este prezentată imaginea obținută prin aplicarea metodei *egalizării histogramei* originalului din figura 3.

Se observă accentuarea contrastului ce permite o mai clară observare a detaliilor, punerea în evidență a variații-

lor minore dintre regiuni care trec neobservate în imaginea inițială.

Concret, metoda egalizării histogramei constă în înlocuirea fiecărui nivel de gri k din imaginea originală cu valoarea frecvențelor cumulate până la acest nivel (valoare scalată, bineînțeles, în intervalul 0 ... 255).

$$k_{nou} = \frac{255}{H_{total}} \sum_{i=0}^{k} h(i), \text{ unde } H_{total} = \sum_{i=0}^{255} h(i)$$

Trebuie notat faptul că egalizarea histogramei poate fi realizată doar pentru o anumită regiune de interes și nu pentru toată imaginea (de exemplu, o zonă determinată de un dreptunghi).

Transformări locale

Aceste transformări țin cont de o anumită vecinătate a pixelului al cărui nivel de gri va fi modificat. De obicei, vecinătățile cu care se lucrează sunt pătratice, de dimensiuni 3×3, 5×5, 7×7, în general de forma $(2p+1) \times (2p+1)$, centrate pe pixelul supus transformării.

Transformările locale, în funcție de scopul lor, se pot clasifica în două categorii:

- transformări care vizează *netezirea imaginii* (*image smoothing*) sau a unor regiuni din aceasta, având drept scop reducerea zgomotului sau a altor defecte ale imaginii;
- transformări care vizează *detecția muchiilor* (*edge detection*), bazate pe derivatele funcției asociată imaginii.

O altă clasificare a transformărilor locale are în vedere proprietățile de liniaritate ale acestora. Astfel, putem vorbi despre ***transformări liniare*** și ***transformări neliniare***.

Transformările liniare se caracterizează prin faptul că valoarea rezultată reprezintă o combinație liniară a nivelurilor de gri din vecinătatea utilizată pentru fiecare pixel. Contribuția fiecărui pixel din vecinătate în rezultatul final este ponderată cu o anumită valoare, astfel:

$$f^{out}_{i^0, j^0} = \sum_{i=-m}^{m} \sum_{j=-m}^{m} w_{i+m, j+m} \cdot f_{i^0+i, j^0+j}$$

unde (i^0, j^0) sunt coordonatele pixelului asupra căruia se realizează transformarea, f reprezintă imaginea inițială iar, f^{out} imaginea finală.

Ecuația de mai sus poate fi scrisă ca un ***produs de convoluție*** dintre imaginea originală f și nucleul (matricea) w, astfel:

$$f^{out} = w \otimes f, \quad w = (w_{ij})_{(2m+1)\times(2m+1)}, \quad f = (f_{ij})_{M \times N}$$

Reducerea zgomotului unei imagini

Metodele privind reducerea zgomotului prezent în imagine (*filtrare*) pot fi liniare sau neliniare, în acord cu discuțiile de mai sus.

Filtrarea liniară se bazează pe înlocuirea nivelului de gri al unui pixel cu media (ponderată sau nu) a valorilor pixelilor din vecinătatea sa.

O metodă simplă care poate rezolva acest deziderat este utilizarea unor nuclee de convoluție de tipul:

$$w_1 = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad w_2 = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad w_3 = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Se observă că prin aplicarea nucleului w_1 imaginii originale, valoarea fiecărui pixel este înlocuită cu media valorilor pixelilor dintr-o vecinătate 3×3 centrată pe pixelul respectiv. Desigur, pot fi folosite vecinătăți de dimensiuni mai mari: 5×5, 7×7 etc. În cazul celorlalte două exemple se atribuie o pondere mai mare pixelului central în calculul mediei, respectiv pixelilor din vecinătatea N_4.

O altă variantă este utilizarea unor nuclee ce reprezintă aproximări ale suprafeței lui *Gauss*, date de funcția:

$$G(x, y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

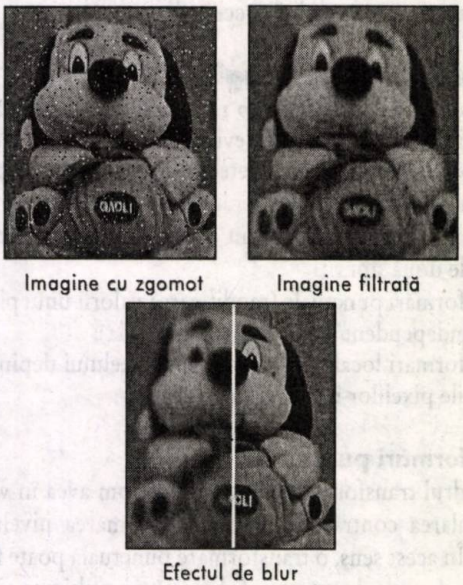
Exemple de nuclee de diferite dimensiuni și pentru valori diferite ale deviației σ sunt date în continuare:

$$w_4 = \frac{1}{32}\begin{bmatrix} 1 & 4 & 1 \\ 4 & 12 & 4 \\ 1 & 4 & 1 \end{bmatrix}, \sigma = 0.391, \quad w_5 = \frac{1}{121}\begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 7 & 11 & 7 & 2 \\ 3 & 11 & 17 & 11 & 3 \\ 2 & 7 & 11 & 7 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}, \sigma = 0.625$$

Se observă ponderarea diferită a pixelilor din vecinătate în funcție de poziția lor față de pixelul central.

Trebuie remarcat faptul că nucleele prezentate mai sus pot fi aplicate consecutiv imaginii, însă o aplicare excesivă conduce la efectul denumit ***blur*** (ceață) și va determina pierderea detaliilor.

În figura 6 este prezentată o operație de filtrare a imaginii din figura 3 la care s-a adăugat un zgomot reprezentat de un număr de pixeli albi și negri generați aleator. A fost utilizat de două ori nucleul w_1 de dimensiune 3×3. Ultima imagine prezintă efectul de "blur" prin aplicarea nucleului w_5 de patru ori succesiv pe jumătatea dreaptă a imaginii originale din figura 3.



Imagine cu zgomot | Imagine filtrată

Efectul de blur

Figura 6

O metodă de filtrare neliniară este dată de filtrajul median al lui *Tuckey*, care constă în înlocuirea valorii fiecărui pixel cu valoarea mediană a șirului de valori ale pixelilor din vecinătatea utilizată. Un exemplu este dat în continuare, pixelul central având nivelul de gri 40, iar vecinătatea folosită fiind de dimensiune 3 × 3:

10	1	12	20	50
100	1	50	30	11
25	20	**40**	50	12
25	25	30	40	50
20	30	50	51	60

Șirul de valori sortat al nivelurilor de gri din vecinătatea considerată este: 1, 20, 25, 30, 30, 40, 40, 50, 50, iar valoarea mediană este 30. Aceasta va fi noua valoare a pixelului central.

Detecția muchiilor

Intuitiv, într-o imagine, muchiile pot fi identificate acolo unde valorile pixelilor se schimbă brusc. În figura 7 sunt prezentate două profile ideale de muchii cât și un caz real de muchie însoțită de zgomot.

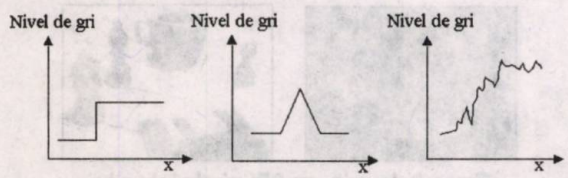


Figura 7: Muchie tip treaptă, muchie tip vârf și muchie reală cu zgomot

Muchiile de tip treaptă se situează între regiunile caracterizate de valori medii ale nivelurilor de gri diferite. Muchiile de tip vârf corespund unei variații locale a intensității pixelilor, variație care va avea un maxim sau un minim.

Schimbările de intensitate pot fi detectate aplicând imaginii un operator de tip *gradient*. *Gradientul* unei funcții într-un punct este un vector normal la curba descrisă de funcție în punctul respectiv. Privind imaginea ca o funcție de două variabile $f(x, y)$, care asociază fiecărui punct (x, y) o valoare a intensității (nivel de gri) și făcând presupunerea că funcția este continuă și derivabilă, gradientul în punctul (x, y) va fi:

$$\nabla f(x,y) = \left(\frac{\partial f}{\partial x}(x,y), \frac{\partial f}{\partial y}(x,y) \right)$$

Amplitudinea și *orientarea* gradientului în punctul (x, y) sunt date de:

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f}{\partial x}(x,y)\right)^2 + \left(\frac{\partial f}{\partial y}(x,y)\right)^2}$$

$$\varphi = \arctan\left(\frac{\partial f}{\partial x}(x,y) \Big/ \frac{\partial f}{\partial y}(x,y) \right)$$

Amplitudinea gradientului măsoară diferența dintre intensitățile pixelilor dintr-o vecinătate, iar *orientarea* relevă direcția celei mai semnificative schimbări de intensitate care este probabil și direcția muchiei.

Aceste formule se transformă în cazul discret al procesării imaginilor digitale prin înlocuirea derivatelor parțiale cu diferențe finite, astfel:

$$\frac{\partial f}{\partial x}(x,y) \approx f(x+1,y) - f(x,y)$$

$$\frac{\partial f}{\partial y}(x,y) \approx f(x,y+1) - f(x,y)$$

În continuare sunt prezentați diferiți operatori bazați pe gradient.

Operatorul Roberts

Acest operator presupune calculul următoarelor diferențe finite:

$$\Delta_1 = f(x,y) - f(x+1,y+1)$$

$$\Delta_2 = f(x,y+1) - f(x+1,y)$$

Amplitudinea și orientarea se calculează în conformitate cu definițiile de mai sus.

Calculul diferențelor finite se poate realiza ca un produs de convoluție între imaginea f și nucleele:

$$w_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad w_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Un dezavantaj al acestui operator îl reprezintă sensibilitatea la zgomot, datorită dimensiunilor mici ale vecinătății folosite.

Operatorul Prewitt

Operatorul *Prewitt* folosește următoarea aproximare a derivatelor în cele două direcții:

$$\frac{\partial f}{\partial x} \approx \frac{1}{2}(f(x+1,y) - f(x-1,y))$$

$$\frac{\partial f}{\partial y} \approx \frac{1}{2}(f(x,y+1) - f(x,y-1))$$

Ca urmare, nucleele folosite în cazul convoluției sunt:

$$w_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad w_2 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} \quad w_3 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \ldots \quad w_8 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{bmatrix}$$

Gradientul este estimat în opt direcții posibile (în cazul unui nucleu 3 × 3) iar valoarea amplitudinii reprezintă maximul rezultatelor obținute prin aplicarea celor opt nuclee. Direcția gradientului se calculează pentru situația corespunzătoare maximului amplitudinii.

Operatorul Sobel

În acest caz, nucleele de convoluție sunt:

$$w_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad w_2 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \quad \ldots \quad w_8 = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix}$$

Se pot folosi toate cele 8 nuclee corespunzătoare celor 8 direcții, într-o manieră similară operatorului *Prewitt*, însă se pot folosi doar nucleele w_1 și w_3 pentru realizarea diferențelor finite pe linii, respectiv coloane. În acest caz, am-

plitudinea și direcția gradientului se calculează în maniera obișnuită:

$$A = \sqrt{w_1^2 + w_3^2} \quad \varphi = \arctan(w_1 / w_3)$$

Există și alți operatori care utilizează metode derivative pentru detecția muchiilor pe care doar îi vom aminti și vom prezenta primul nucleu de convoluție (celelalte pot fi obținute prin rotații succesive):

$$\begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \end{bmatrix}$$

Operatorii Kirsch, Robinson și Frei & Chen

Trebuie ținut cont de faptul că numai o valoare mare a amplitudinii gradientului poate semnifica prezența unei muchii. Astfel, se impune o filtrare a rezultatelor obținute (de exemplu, se păstrează ca potențiale componente ale muchiei acei pixeli ale căror valori depășesc un anumit prag).

Figura 8 prezintă un exemplu de aplicare al unui operator de detecție a muchiilor (*Prewitt*) asupra imaginii din figura 3:

Figura 8

Alte metode de detecție a muchiilor se bazează pe aplicarea operatorului *Laplace*, ținându-se cont de derivata a doua a funcției asociată imaginii:

$$L(x,y) = \frac{\partial^2 f}{\partial x^2}(x,y) + \frac{\partial^2 f}{\partial y^2}(x,y)$$

Dacă în cazul precedent existența unei muchii era determinată de un maxim al gradientului (bazat pe derivate de ordin I), în acest caz muchiile sunt identificate prin trecerea prin 0 a *laplacian*-ului imaginii (un maxim al derivatei de ordinul I reprezintă un zero al derivatei de ordinul II).

Laplacian-ul poate fi calculat printr-un produs de convoluție al imaginii cu nuclee de forma:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \text{ sau } \begin{bmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{bmatrix}$$

Alte abordări privind detecția muchiilor au în vedere operatori de tip *DoG* (*Difference of Gaussians*) sau *LoG* (*Laplacian of a Gaussian*).

Tehnici de segmentare

Prin segmentare se înțelege divizarea imaginii într-un număr finit de regiuni de interes, în conformitate cu anumite criterii. O segmentare completă a unei imagini (privită ca o regiune globală R) constă în determinarea unui set de regiuni $R_1, R_2, \ldots R_p$, astfel încât să fie îndeplinite condițiile:

$$R = \bigcup_{i=1}^{p} R_i \,,\; R_i \bigcap R_j = \varnothing,\; \forall i, j \in \{1, 2, \ldots, p\}, i \neq j$$

Un exemplu de segmentare îl constituie separarea obiectelor prezente în imaginea de fundal. În cazul în care atât obiectele cât și fundalul se caracterizează prin niveluri medii de gri diferite, acest lucru se poate realiza printr-o simplă filtrare a nivelurilor de gri în funcție de un anumit prag T (*threshold*), rezultând o imagine binară f^{out} astfel:

$$f_{i,j}^{out} = \begin{cases} 1, & \text{dacă } f_{i,j} \leq T \\ 0, & \text{dacă } f_{i,j} > T \end{cases}$$

În figura 9 este prezentată o imagine cu 256 niveluri de gri și rezultatul unui proces de segmentare, valoarea 0 codificând fundalul, iar valoarea 1 obiectele:

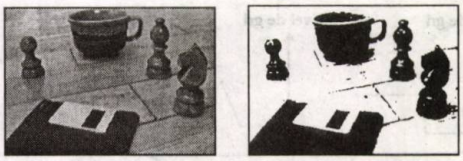


Figura 9: Imagine cu 256 niveluri de gri și imagine segmentată T=110

Trebuie notat faptul că performanțele segmentării sunt determinate de alegerea nivelului de prag T. Metodele privind alegerea valorii acestuia iau în considerare histograma nivelurilor de gri a imaginii. Pentru cazul considerat, histograma este prezentată în figura 10.

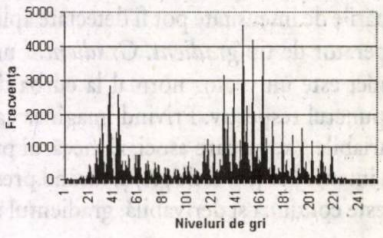


Figura 10

Histograma imaginii este bimodală în sensul că există două concentrări importante ale nivelurilor de gri: în jurul valorii 41 (nivelul mediu de gri al obiectelor) și în jurul valorii 150 (nivelul mediu de gri al fundalului). Ca urmare, nivelul pragului poate fi ales între cele două valori. (T = 110).

Un algoritm simplu iterativ pentru alegerea automată a pragului în cazul unei histograme bimodale presupune următorii pași:

- alegerea unei prime aproximări pentru T (de exemplu T = 1);
- la pasul i se calculează nivelurile medii de gri ale obiectelor și fundalului:

nivel mediu gri obiecte $\bar{n}_{obiecte} = \frac{\sum_{i,j} f_{i,j}}{total\ pixeli\ obiecte}, f_{i,j} \le T$

nivel mediu gri fundal $\bar{n}_{fundal} = \frac{\sum_{i,j} f_{i,j}}{total\ pixeli\ fundal}, f_{i,j} > T$

- calcularea pragului T la pasul i:

$$T = \frac{\bar{n}_{obiecte} + \bar{n}_{fundal}}{2}$$

- dacă valoarea T a pragului la pasul i este egală cu valoarea pragului la pasul anterior sau dacă a expirat numărul de iterații dorite, algoritmul se încheie, altfel se reia cu pasul 2.

De regulă, un număr de iterații între 4 și 10 este suficient. În cazul exemplului prezentat în figura 9, cu un număr de 10 iterații s-a obținut o valoare de prag $T = 119$.

În cazul histogramelor multimodale (de exemplu, pot exista obiecte cu niveluri medii de gri diferite), pot fi alese mai multe niveluri de prag, corespunzătoare concentrării nivelurilor de gri în jurul unor anumite puncte în histograma imaginii:

$$f_{i,j}^{out} = \begin{cases} 1, & \text{dacă } f_{i,j} \in [T_0,T_1] \cup [T_2,T_3] \cup \ldots \cup [T_{r-1},T_r] \\ 0, & \text{altfel} \end{cases}$$

În cazul imaginilor color, segmentarea presupune calcularea histogramelor pentru cele trei culori de bază (R, G, B) și determinarea pragurilor T în funcție de culoarea obiectelor care se doresc a fi separate de fundal. Se realizează segmentarea pentru fiecare componentă, rezultând trei imagini binare finale care vor fi supuse unei operații de tip *OR*.

Analiza și recunoașterea formelor

Procesul de segmentare conduce la obținerea unei imagini binare care conține obiectele separate de fundal. Utilizând un algoritm de determinare a componentelor conexe se realizează distingerea și etichetarea obiectelor, ceea ce creează premisele pentru ultima etapă în cadrul procesării imaginilor: analiza formelor. În urma acestei analize, pentru fiecare obiect în parte se pot calcula diverși parametri (cum ar fi invarianții), parametri care pot fi utilizați într-un sistem automat de clasificare. Există o graniță foarte fină în această etapă între domeniul procesării imaginilor și domeniul recunoașterii formelor. Prin *formă* se înțelege reprezentarea matematică a unui obiect pe baza observațiilor și măsurătorilor efectuate asupra lui.

Există două direcții principale în domeniul analizei formelor, și anume:

- abordarea funcțională;
- abordarea din perspectiva teoriei mulțimilor.

Fiecare dintre cele două direcții definește conceptul de formă, plecând de la diferite abordări matematice.

Abordarea funcțională se bazează pe aproximarea contururilor formelor (obținute eventual printr-o metodă de detecție a muchiilor) cu diverse funcții, ceea ce permite aplicarea rezultatelor geometriei diferențiale și analitice în studiul acestora. Funcțiile contur rezultate pot fi aproximate cu serii trigonometrice de funcții *sin* și *cos*, ceea ce permite o caracterizare a formelor cu ajutorul coeficienților *Fourier*. Abordarea din perspectiva teoriei mulțimilor privește formele ca fiind mulțimi compacte și închise și are în vedere obținerea anumitor proprietăți geometrice sau rapoarte. Se pot calcula următoarele proprietăți:

- $A(X)$ - aria formei X;
- $P(X)$ - perimetrul formei X;
- $D_A(X)$, $D_P(X)$ - diametrele cercurilor având aceeași arie, respectiv perimetru cu forma X;
- $FMD(X)$ - *Feret Maximum Diameter* - cea mai mare distanță dintre două puncte aparținând formei X (este o măsură a alungirii);
- $C(X)$ - circularitatea $C(X) = 4\pi A(X)/(P(X))^2$ (caracterizează devierea formei X față de un disc; valoarea maximă este 1 în cazul discului);
- $a(X)/b(X)$ - raportul lungimilor axelor unei elipse având aceeași arie și același perimetru cu forma X (acest raport este o măsură a alungirii);
- Raportul de convexitate: $A(X)/A(conv(X))$, unde $conv(X)$ reprezintă poligonul dat de înfășurătoarea convexă a formei X (raportul este o măsură a convexității, pentru o formă convexă luând valoarea 1, iar în caz contrar valoarea este subunitară).

Au fost prezentate numai câteva rapoarte și proprietăți doar pentru exemplificare, alegerea și definirea lor făcându-se în funcție de specificul fiecărei aplicații. O proprietate importantă pe care parametrii pot să o dețină este invarianța (la translație, rotație, scalare), un exemplu în acest sens fiind momentele de diferite ordine.

În final, o formă va putea fi caracterizată de un set de valori (x_1, x_2, ..., x_p), urmând aplicarea tehnicilor specifice recunoașterii formelor în vederea clasificării acesteia.

Concluzii

Articolul s-a dorit a fi o introducere sumară în metodele și tehnicile folosite în cadrul procesării imaginilor, multe aspecte fiind ignorate (cum ar fi: reconstrucția imaginilor, compresia, analiza texturii, reconstrucția *3D* etc.). Aplicațiile acestui domeniu sunt practic nelimitate: de la simple corecții ale defectelor imaginilor până la aplicații complexe în medicină, industrie, securitate, armată.

Bibliografie

1. Pratt W.K., *Digital Image Processing, 2nd Edition*, John Wiley & Sons, New York, 1991.
2. Russ J.C., *The Image Processing Handbook, 2nd Edition*, CRC Press.
3. Cocquerez J.P., Philipp S., *Analyze d'Images: Filtrage et Segmentation.*
4. Marion A., *Introduction aux Tehniques de Traitement d'Images.*
5. Toumazet J.J., *Traitement de l'Image sur Micro-ordinateur.*

Radu-Daniel Vatavu este student în anul III la Universitatea Ștefan cel Mare din Suceava. Poate fi contactat prin e-mail la **raduvro@yahoo.com**.

Războiul browserelor...

Generația a ȘASEA

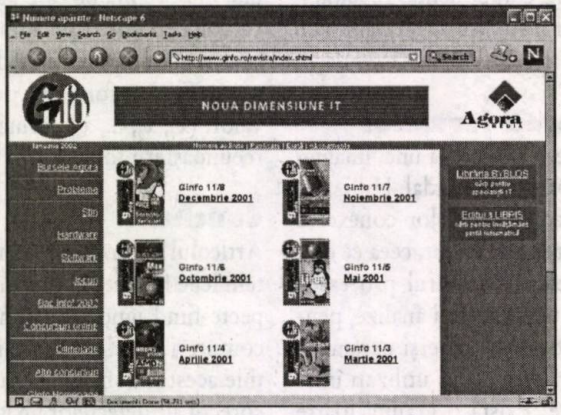
Mihai Scorțaru

În cursul anului 2001 au fost lansate pe piață versiunile cu numărul 6 ale celor mai populare programe de navigare: Netscape Navigator, Internet Explorer și Opera. În cadrul acestui articol vom realiza o scurtă prezentare a fiecărui browser, precum și o prezentare comparativă a celor trei browsere.

Netscape Navigator

După explozia Internetului de la începutul anilor '90, *Netscape Navigator* a fost cel mai folosit program de navigare. Calitățile sale au fost unanim recunoscute, iar produsele concurente aveau puține șanse să amenințe poziția sa de lider.

După anul 1995, compania rivală *Microsoft* și-a dat seama că poziția sa pe piață depinde foarte mult de serviciile pentru navigarea pe Internet oferite și a început războiul browserelor. În acel moment *Netscape Navigator* ajunsese la versiunea 3 și încă domina piața.



În scurt timp a fost lansată versiunea *Netscape* 4, dar aceasta a pierdut din avantajul pe care îl avea în fața rivalului său *Microsoft Internet Explorer*. După integrarea browserului *Internet Explorer* în sistemele de operare *Windows*, lupta nu a mai fost egală. *Netscape* a pierdut fără drept de apel poziția de lider pe piața programelor de navigare.

Mai mult, *Internet Explorer* era un produs gratuit, motiv pentru care compania *Netscape* a fost nevoită să ofere și ea gratuit browserul propriu. Pentru a contracara ofensiva inițiată de concurență, *Netscape* a încercat să își îmbunătățească produsele. Astfel a apărut *Netscape Communicator*, un pachet de programe care cuprindea, pe lângă browser, un client de e-mail, un editor de pagini *Web* etc. Ultima versiune a acestui produs, *Netscape Communicator* 4.78, a fost lansată la sfârșitul anului 2001.

Lucrul la versiunea 5 a browserului a început acum câțiva ani. Pentru a crea un program de navigare performant, *Netscape* a apelat la comunitatea *open source*. Astfel, codul sursă al noului browser a devenit public și oricine putea contribui la realizarea programului de navigare.

După amânări succesive se părea că noul *Netscape Navigator* este pregătit pentru lansare. Surpriza a venit atunci când a fost anunțat faptul că noua versiune va avea numărul 6. Astfel, se evita dezavantajul psihologic pe care l-ar fi avut *Netscape* 5 față de *Internet Explorer* 6. Nu a existat nici o versiune oficială a produsului care să poarte denumirea *Netscape* 5.

Netscape 6 a dezamăgit imediat după lansare. Browserul era foarte instabil și practic inutilizabil. În plus, era un mare consumator de resurse hardware. După câteva săptămâni a apărut versiunea 6.01 care rezolva câteva dintre problemele de stabilitate, dar browserul era departe de așteptările utilizatorilor.

Versiunea 6.1 a adus îmbunătățiri majore, dar chiar și acest browser avea probleme de stabilitate. *Netscape* susține că acestea au fost complet eliminate în versiunea 6.2, lansată la sfârșitul anului trecut. Versiunea curentă este 6.21, dar diferențele față de *Netscape* 6.2 sunt neesențiale.

Caracteristici

Netscape 6 este un produs care poate fi adaptat dorințelor fiecărui utilizator. Interfața este *skin*-abilă ceea ce permite utilizatorilor să o modifice în funcție de preferințele proprii.

Produsul oferă un suport foarte bun pentru ultimele standarde *Web* cum ar fi *HTML* 4.01, *XML*, *CSS*, *JavaScript* etc., dar acesta poate fi îmbunătățit.

Din nefericire, *Netscape* 6 nu este perfect compatibil cu *site*-urile proiectate pentru *Internet Explorer*, mai are unele probleme de stabilitate și performanțele sale nu se ridică la nivelul așteptărilor.

Programul de navigare *Netscape* este disponibil în variante pentru diverse sisteme de operare cum ar fi *Windows*, *Linux*, *MacOS* etc.

Internet Explorer

Pentru a încerca să acapareze o parte din piața dominată de *Netscape*, *Microsoft* a început dezvoltarea propriului său program de navigare. Totuși, poziția companiei rivale era destul de greu de zdruncinat, motiv pentru care *Microsoft* a trebuit să aleagă o soluție extremă pentru a impune pe piață propriul program de navigare.

După trecerea de la versiunea 3 la versiunea 4, *Microsoft* a decis ca *Internet Explorer* să fie inclus în distribuția standard a sistemului de operare *Windows* fără a exista posibilitatea de a fi dezinstalat.

Astfel, majoritatea utilizatorilor au preferat să nu instaleze alte browsere pentru a economisi spațiu. Mișcarea a avut succes și browserul a devenit în scurt timp lider al pieței.

Avantajul decisiv a fost obținut după lansarea versiunii *Internet Explorer* 5. Performanțele excelente ale acestui browser l-au făcut să acapareze peste 80% din piață.

Au urmat versiunile *Internet Explorer* 5.01 și *Internet Explorer* 5.5 care nu au adus modificări majore.

Ultima versiune a fost lansată în luna august a anului 2001 și a întărit poziția de lider a produsului care deține acum aproape 90% din piață.

Caracteristici

Internet Explorer 6 este cel mai cunoscut program de navigare disponibil în acest moment pe piață. El oferă cel mai bun suport pentru ultimele tehnologii Web, chiar dacă acesta nu este complet. De exemplu, specificația *CSS*-2 nu este integral implementată.

Singurul dezavantaj în fața concurenților săi îl reprezintă faptul că interfața sa nu poate fi personalizată, neexistând posibilitatea folosirii unor *skin*-uri.

Internet Explorer 6 este inclus în sistemele de operare *Microsoft Windows XP Home Edition* și *Microsoft Windows XP Professional*, dar poate fi folosit și împreună cu alte sisteme de operare *Windows* sau *MacIntosh*.

Este cunoscut faptul că *Microsoft* nu oferă suport pentru sistemul de operare rival: *Linux*. Din acest motiv nu există variante decât pentru sistemele de operare *Windows* și *MacOS*.

Opera

Programul de navigare *Opera* a început să facă mici pași pe piața browserelor imediat după lansare datorită dimensiunii reduse și a vitezei de încărcare a paginilor.

Deși performanțele acestui mic browser erau extraordinare, el nu a reușit să se impună pe piață datorită faptului că nu oferea un suport suficient pentru tehnologiile *Web* și nu era gratuit.

Situația a început să se schimbe în momentul apariției versiunii *Opera* 5. Acesta a fost primul browser *Opera* oferit gratuit. Totuși, pentru a putea beneficia de facilitățile sale, utilizatorii erau obligați să vizualizeze o reclamă afișată permanent în fereastra aplicației. Aceasta putea fi eliminată dacă se achiziționa, contra cost, versiunea fără reclame a programului de navigare.

Suportul pentru tehnologiile *Web* a fost îmbunătățit, dar existau o mulțime de pagini care nu puteau fi afișate corect de *Opera* în timp ce *Netscape Navigator* și *Internet Explorer* nu aveau probleme la afișarea lor.

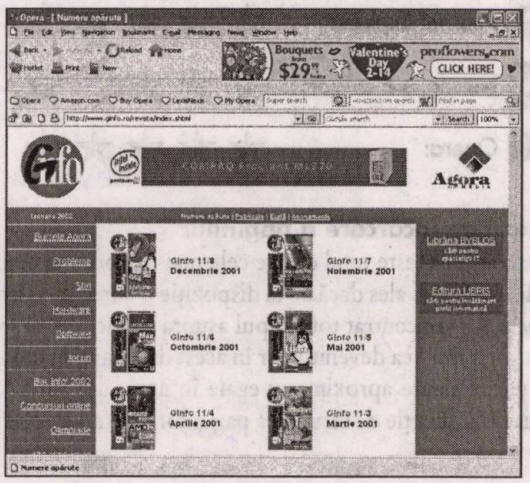
Situația a început să se schimbe începând cu versiunea 6, dar nici *Opera* 6 sau *Opera* 6.01 nu oferă suport complet pentru ultimele tehnologii cum ar fi *CSS*, *XML* etc.

Caracteristici

Ultima versiune a programului de navigare *Opera* introduce o nouă interfață grafică care, dacă utilizatorul dorește acest lucru, o poate înlocui pe cea a versiunilor 5.0, 5.1, 5.11 sau 5.12.

Aceasta este *skin*-abilă, dar *skin*-urile nu pot fi instalate automat ca în cazul programului de navigare *Netscape* 6. Din nefericire, există unele *bug*-uri în această interfață care duc la obținerea unor efecte neplăcute pentru utilizator. De asemenea, produsul nu este foarte stabil, programul blocându-se destul de des.

Browserul *Opera* este disponibil în variante pentru diverse sisteme de operare cum ar fi *Windows*, *Linux* etc.



Prezentare comparativă

În cele ce urmează vom compara cele trei programe de navigare în funcție de diverse criterii și vom încerca să determinăm performanțele acestora din mai multe puncte de vedere.

Dimensiunea programului

Cel mai mic browser este, fără îndoială, *Opera*. Dimensiunea programului de instalare pentru *Opera* 6 este de doar 3.2 MB, dacă nu se dorește și instalarea mașinii virtuale *Java*. Programul de navigare permite folosirea oricărei mașini virtuale *Java* care este instalată în sistem. Dacă se dorește instalarea împreună cu o mașină virtuală *Java*, dimensiunea programului de instalare depășește cu puțin 10 MB.

Internet Explorer și *Netscape Navigator* permit descărcarea unui program mic cu dimensiunea de aproximativ 500 KB care permite alegerea componentelor care se doresc a fi instalate.

În cazul browserului *Netscape Navigator*, dimensiunea totală a fișierelor care sunt descărcate este de aproximativ 21 MB.

Dimensiunea totală a fișierelor pentru *Internet Explorer* depinde foarte mult de opțiunile alese, dar este de cel puțin 10 MB.

Internet Explorer: ★★★★☆

Netscape Navigator: ★★★☆☆

Opera: ★★★★★

Viteza de lansare în execuție

Din acest puncte de vedere, *Internet Explorer* are un avantaj important. O parte din modulele necesare sunt încărcate automat după lansarea sistemului de operare *Windows*.

Cu toate acestea, *Opera* se încarcă mult mai repede datorită dimensiunii reduse a programului. *Netscape* nu beneficiază de nici unul dintre avantajele concurenților săi, motiv pentru care este cel mai lent la încărcare.

Internet Explorer: ★★★★☆

Netscape Navigator: ★★★☆☆

Opera: ★★★★★

Viteza de încărcare a paginilor

Acest criteriu este unul dintre cele mai importante pentru utilizatori, mai ales dacă au la dispoziție o conexiune lentă. *Opera* s-a concentrat tot timpul asupra vitezei de încărcare a paginilor și a devenit lider în acest domeniu. Rivalii săi au performanțe aproximativ egale în acest domeniu, dar variază în funcție de structura paginilor care sunt încărcate.

Internet Explorer: ★★★★☆

Netscape Navigator: ★★★★☆

Opera: ★★★★★

Suportul tehnologiilor Web

Din acest punct de vedere *Internet Explorer* se află în fața competitorilor. Aceasta se datorează faptului că *Microsoft* are un cuvânt greu de spus în definirea diferitelor standarde. *Nestcape* implementează o mare parte a acestor standarde, dar există anumite pagini care nu sunt afișate corect. Suportul oferit de *Opera* nu se ridică la nivelul competitorilor chiar dacă a fost îmbunătățit semnificativ la ultima versiune. Majoritatea paginilor care folosesc tehnologii mai noi nu sunt corect afișate de către *Opera*.

Internet Explorer: ★★★★☆

Netscape Navigator: ★★★★☆

Opera: ★★☆☆☆

Stabilitate

Un alt factor important îl reprezintă stabilitatea programului. *Internet Explorer* nu are probleme de acest tip, programul fiind foarte stabil. Dificultățile întâmpinate de *Netscape* după lansarea versiunii 6.0 au fost în mare parte depășite la versiunile 6.1 și 6.2. *Opera* nu avea probleme de stabilitate până la versiunea 5, dar *Opera* 6 îngheață destul de des și duce deseori chiar la blocarea sistemului.

Internet Explorer: ★★★★☆

Netscape Navigator: ★★★☆☆

Opera: ★★★☆☆

Securitate

Securitatea a fost întotdeauna punctul forte al programului de navigare *Netscape*. Chiar dacă aceasta nu este perfectă, *Netscape* este lider în acest domeniu. *Internet Explorer* s-a confruntat întotdeauna cu probleme de securitate care erau remediate prin lansarea unor *patch*-uri. Nici versiunea 6 nu a scăpat de această problemă; la puțin timp după lansare au fost descoperite unele modalități de a înlătura protecțiile oferite de acest produs. *Opera* și-a îmbunătățit constant performanțele în acest domeniu și oferă acum o securitate destul de bună.

Internet Explorer: ★★★★☆

Netscape Navigator: ★★★★★

Opera: ★★★☆☆

Afișarea diacriticelor

Pentru cei care vizitează pagini redactate în limba română, o importanță destul de mare o are modul în care sunt afișate diacriticele. Dacă documentul *HTML* încărcat nu indică setul de caractere care trebuie folosit, atunci programul de navigare va folosi setul de caractere implicit. Din aceste motive, pentru afișarea corectă a diacriticelor trebuie ales ca set de caractere implicit standardul *ISO*-8859-2. Fiecare program de navigare permite alegerea setului de caractere, dar în *Opera* această opțiune este destul de greu de găsit. Celelalte produse nu au probleme din acest punct de vedere.

Internet Explorer: ★★★★★

Netscape Navigator: ★★★★★

Opera: ★★☆☆☆

Interfața grafică

Acest criteriu este foarte important pentru unii utilizatori. Interfețele grafice ale majorității programelor pot fi personalizate pentru a satisface preferințele utilizatorilor. Doar *Internet Explorer* nu oferă o interfață *skin*-abilă, dar acest neajuns este compensat cu ușurința de utilizare a interfeței standard. Noua interfață grafică oferită de *Opera* 6 poate duce la confuzii deoarece pot fi deschise mai multe ferestre, dar fiecare dintre acestea pot conține mai multe pagini. După ce vă veți obișnui, acest lucru nu vă va mai deranja.

Internet Explorer: ★★★☆☆

Netscape Navigator: ★★★★★

Opera: ★★★★☆

Ușurința în utilizare

Modul în care se realizează navigarea are o importanță deosebită. Ușurința cu care se utilizează sistemul depinde foarte mult de interfața grafică, dar nu numai. Din acest punct de vedere, *Internet Explorer* are cea mai intuitivă interfață. *Opera* se confruntă cu aceleași probleme datorită confuziilor care pot apărea atunci când există mai multe ferestre care conțin mai multe pagini deschise.

Internet Explorer: ★★★★★

Netscape Navigator: ★★★★☆

Opera: ★★★★☆

Apreciere generală

Pentru a realiza o comparare în ansamblu a celor trei produse, vom lua în considerare toate criteriilor prezentate anterior. Evident, fiecare va avea o pondere diferită, în funcție de importanța sa. Am ales ponderile în funcție de părerile noastre care pot fi diferite de ale dumneavoastră; dacă credeți că unui anumit criteriu i s-a acordat o importanță prea mică sau prea mare, puteți modifica ponderea acestuia și recalcula punctajele finale. Ponderile alese de noi sunt:

- Dimensiunea: 10% (*d*: 2)
- Viteza de lansare în execuție: 10% (*vl*: 2)
- Viteza de încărcare a paginilor: 25% (*vi*: 5)
- Suportul tehnologiilor Web: 10% (*sw*: 2)
- Stabilitatea: 10% (*st*: 2)
- Securitatea: 15% (*se*: 3)
- Afișarea diacriticelor: 5% (*dia*: 1)
- Interfața grafică: 5% (*ig*: 1)
- Ușurința în utilizare: 10% (*uu*: 2)

Pentru a obține punctajele finale, vom realiza o medie geometrică în care fiecare criteriu va avea importanța indicată mai sus. Formula folosită este:

$$punctaj = \sqrt[20]{d^2 \cdot vl^2 \cdot vi^5 \cdot sw^2 \cdot st^2 \cdot se^3 \cdot dia \cdot ig \cdot uu^2}\ .$$

Punctajele obținute sunt:

Internet Explorer: **3.84**

Netscape Navigator: **3.61**

Opera: **3.45**

Concluzii

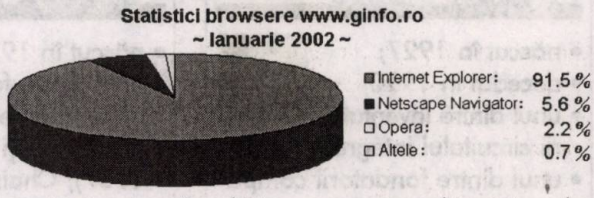
Internet Explorer își justifică poziția de lider al pieței, obținând cel mai mare punctaj. Totuși, competitorii săi se află la o distanță destul de mică, ceea ce nu explică segmentul foarte mare de piață deținut de *Internet Explorer*.

Opera este un browser promițător care ar putea deveni preferatul multora dacă sunt îmbunătățite stabilitatea, suportul pentru tehnologiile *Web* și interfața grafică.

Netscape Navigator nu reușește să detroneze *Internet Explorer*, performanțele sale fiind doar cu puțin superioare celor demonstrate de *Opera*.

Datorită politicii agresive aplicate de *Microsoft* pentru impunerea propriului program de navigare, *Internet Explorer* deține cel mai mare segment de piață în acest moment. Urmează *Netscape Navigator* la o distanță foarte mare; *Opera* are un potențial ridicat, segmentul său de piață crescând permanent.

În imaginea următoare puteți vedea statistica browserelor folosite de către cei care au accesat *site*-ul revistei noastre în cursul lunii ianuarie.



Am încercat să realizăm o comparare obiectivă a celor trei programe de navigare care să vă ajute să îl alegeți pe cel potrivit nevoilor dumneavoastră. ***Decizia vă aparține!***

INTEL Corporation

Carmen Șușman

Istoria companiei Intel se confundă, în mare măsură, cu istoria microprocesoarelor. Chiar dacă, în ultimii ani, unele produse ale concurenței sunt mai performante, Intel rămâne una dintre companiile care au avut o influență decisivă asupra domeniului IT.

Nașterea companiei

Compania *Intel* a fost creată de *Robert Noyce* și *Gordon Moore*. Ei lucrau pentru compania *Fairchild Camera and Instrument*, primul având funcția de manager general, iar al doilea fiind conducătorul departamentului de cercetare și dezvoltare.

Înainte de a deveni membri fondatori ai companiei *Fairchild*, cei doi erau angajați la *Shockey Semiconductor* și au renunțat la funcțiile lor datorită faptului că nu erau mulțumiți de modul în care era condusă firma.

Ulterior, ei nu au mai fost mulțumiți de felul în care erau gestionate profiturile firmei *Fairchild* și s-au hotărât să repete experiența. Astfel, în iulie 1968, împreună cu *Andy Grove* (un alt angajat de la *Fairchild*), au creat compania *Intel*. Numele este prescurtarea de la *Integrated Electronics*.

Noyce și *Grove* au investit câte 250.000$ pentru a crea noua companie și au reușit să atragă alte investiții în valoare totală de 2.500.000$. Compania avea ca domeniu de activitate producerea unor cipuri de memorie.

Intel urmărea integrarea unui număr mare de tranzistori pe un cip de silicon, folosind un nou proces fotochimic descoperit de *Robert Noyce*. Cei trei ingineri au reușit să construiască primul cip pentru memoriile calculatoarelor, produs care va înlocui produsele bazate pe medii magnetice.

Scopul companiei era de a reduce costul corespunzător unui bit de memorie prin creșterea capacității cipurilor.

Primele produse

În 1969 a apărut primul produs al companiei *Intel*, și anume *3101*, o memorie *Schottky* statică de 64 de bits cu acces aleator (*SRAM*).

După puțin timp a apărut versiunea *1101* a acestui produs, care era tot o memorie *SRAM*, dar de *256* de biți. Acest produs a fost construit folosind o tehnologie numită *silicon gate metal-oxide semiconductor process*.

În 1970 a fost lansat cipul *1103*, prima memorie cu acces aleator dinamică (*DRAM*) produsă de *Intel*, care a avut succes pe piață. Acest cip era fabricat folosind același proces *MOS* (*Metal-Oxide Semiconductor*). La sfârșitul anului 1971 cipul a devenit cel mai bine vândut produs de acest tip din lume.

În 1971 *Intel* a lansat primul cip de memorie *EPROM* (*Erasable Programmable Random Access Memory*). Acest cip, inventat de *Dov Frohman*, oferea posibilitatea stocării permanente a datelor (la fel ca și alte memorii *ROM* disponibile pe piață), dar permitea și ștergerea datelor și posibilitatea refolosirii. Ștergerea se realiza folosind o undă de raze ultraviolete.

Inventarea microprocesorului

În anul 1969, compania japoneză *Busicomp* a cerut companiei *Intel* să proiecteze un set de cipuri pentru o familie de calculatoare programabile.

Proiectul a fost condus de *Marcian Ted Hoff*; el a decis să nu proiecteze 12 cipuri distincte, ci un set de patru

Robert Noyce

- născut în 1927;
- decedat în 1990;
- unul dintre inventatorii circuitului integrat (1958);
- unul dintre fondatorii companiei Intel (1968);
- președinte și CEO al SEMATECH (1988-1990).

Gordon Moore

- născut în 1929;
- unul dintre fondatorii companiei Intel;
- Chairman și CEO (1979-1987); Chairman Emeritus (1997-2002)
- renumit pentru celebra lege care îi poartă numele.

Andy Grove

- născut în 1936;
- al patrulea angajat al companiei Intel;
- a ocupat funcțiile de președinte (1979-1987), Chief Executive Officer (1987-2002) și Chairman (1997-2002).

cipuri a căror funcționare să fie coordonată de un dispozitiv central, care putea fi programat folosind instrucțiuni stocate într-un cip de memorie. Cu ajutorul lui *Stan Mazor*, el a reușit să proiecteze cipul aritmetic *4004*, iar produsul a fost acceptat de *Busicomp*.

Un alt angajat al companiei, *Frederico Faggin*, a preluat proiectul și a început să construiască cipul propriuzis. Astfel, a apărut microprocesorul *4004*, un cip pe 4 biți (putea procesa 4 biți la un moment dat) care conținea 2300 de tranzistori *MOS*.

Intel și-a dat seama de potențialul produsului, dar *Busicomp* avea drepturi de proprietate asupra acestuia. Compania americană le-a oferit japonezilor o investiție de 60.000$ în schimbul drepturilor asupra cipului.

În noiembrie 1971 *Intel* a lansat pe piață noul produs. Microprocesorul avea propriul său set de instrucțiuni și trebuia programat pentru a efectua diferite operații. Publicul și producătorii de calculatoare au fost informați despre posibilitățile oferite de *4004* și a fost descris modul în care poate fi programat acesta.

Cipul *4004* a fost îmbunătățit de către echipa condusă de *Hoff* și, în anul 1972, a fost lansat cipul *8008* cu o integrare sporită, mai rapid și putând procesa 8 biți la un moment dat. Doi ani mai târziu a apărut *8080*, un microprocesor capabil să execute 290.000 de operații pe secundă, o performanță de zece ori mai mare decât cea a predecesorului său. Microprocesorul a devenit "inima" primului calculator personal.

În 1978 a fost lansat *8086*, primul procesor care putea procesa 16 biți simultan. Era de zece ori mai rapid decât cipul *8080* și conținea 27.000 de tranzistori.

Implicarea IBM

În anul 1980 *IBM* a intrat pe piața *PC*-urilor și a decis să folosească microprocesorul *Intel 8080* pentru unitățile centrale ale calculatoarelor sale.

Un an mai târziu, *IBM* a lansat pe piață propriul calculator personal. Datorită reputației acestei companii, produsele *Intel* au fost folosite în întreaga lume.

După puțin timp, mai multe companii au început să producă "clone" compatibile cu modelele *IBM*, ceea ce însemna că toți acești producători foloseau microprocesoare produse de *Intel*.

Ultimii 20 de ani

În ultima perioadă *Intel* a continuat să lanseze procesoare noi. Mai multe detalii despre acestea puteți citi în casetele dedicate microprocesoarelor *Intel*.

Microprocesorul *80286*, lansat în 1982, a ajuns să fie folosit în peste 15 milioane de *PC*-uri în anul 1988.

Istoria microprocesoarelor

Intel 4004 - 1971

Acesta a fost primul microprocesor produs de *Intel*. Cipul *4004* a fost o invenție revoluționară care a deschis calea spre calculatoarele personale.

Intel 8008 - 1972

Cipul *8008* a fost de două ori mai puternic decât predecesorul său și a fost folosit la construirea primului calculator destinat utilizatorilor casnici: *Mark-8*. Potrivit standardelor actuale, acest calculator era foarte greu de produs, de menținut și de folosit.

Intel 8080 - 1974

Cipul *8080* a fost folosit la producerea primului calculator personal: *Altair*. Denumirea acestui calculator a fost inspirată de una dintre destinațiile navei stelare *USS Enterprise* din celebra serie de televiziune *Star Trek*. După lansarea pe piață a acestui calculator, a apărut pentru prima dată situația în care unele comenzi nu au putut fi onorate imediat.

Intel 8088/8086 - 1978

Decizia companiei *IBM* de a se implica în industria calculatoarelor personale a făcut ca acest procesor să devină "creierul" produsului de succes *IBM-PC*. Ca urmare, *Intel* a devenit una dintre cele mai de succes firme ale anilor '70.

Intel 286 - 1982

Cunoscut și sub denumirea de *80286*, acest microprocesor a fost primul care păstra compatibilitatea cu predecesorul său. Astfel aplicațiile scrise pentru *8088/8086* puteau fi rulate și folosind noul procesor. Această caracteristică a fost păstrată pentru toate procesoarele *Intel* lansate ulterior, chiar și procesoarele actuale păstrând compatibilitatea cu cele din generația *8086*.

Intel 386 - 1982

Acest procesor conținea peste 290.000 de tranzistori, de peste zece ori mai mult decât *4004*, primul procesor produs de *Intel*. A fost primul procesor pe 32 de biți și primul care oferea facilități *multitasking* (putea rula simultan mai multe programe).

Intel 486 - 1989

Lansarea acestui procesor a însemnat trecerea de la interfețele cu linie de comandă la cele *point and click*. Era posibilă, pentru prima dată, folosirea unei interfețe grafice în culori și realizarea tehnoredactării asistate de calculator cu o viteză semnificativă. De asemenea, acesta a fost primul procesor care integra, în același cip, coprocesorul matematic. Ca urmare, viteza de efectuare a calculelor s-a mărit semnificativ, deoarece procesorul central nu mai trebuia să efectueze calculele matematice complexe.

Totuși, *Intel* a întâmpinat și dificultăți de-a lungul timpului. La mijlocul anilor '80, competitorii japonezi au început o politică de reducere a prețurilor circuitelor integrate (în special pentru cipuri *DRAM* și *EPROM*). Între 1985 și 1986 *Intel* a fost nevoită să scadă salariile, să renunțe la o treime dintre angajați și să abandoneze unele proiecte.

În 1987 a intervenit guvernul *Statelor Unite* pentru a ajuta companiile americane să iasă din impas. Doisprezece producători importanți din industria semiconductorilor și-au unit forțele și au fondat *SEM ANTECH* (*Semiconductor Manufacturing Technology*) pentru a-și asigura avansul față de rivalii japonezi. Președinte și *CEO* al acestui consorțiu a fost chiar unul dintre fondatorii companiei *Intel*, *Bob Noyce*. El a rămas la conducerea consorțiului până în 1990, anul morții sale.

Primul microprocesor pe 32 de biți a fost *386*, introdus în anul 1985. El conținea 275.000 de tranzistori și executa cinci milioane de instrucțiuni pe secundă.

Au urmat alte microprocesoare pe 32 biți cum ar fi *486*, *Pentium*, *Pentium Pro*, *Pentium II*, *Pentium II Xeon*, *Celeron*, *Pentium III*, *Pentium III Xeon*, *Pentium 4* și *Xeon*.

Anul trecut a fost lansat *Intel Itanium*, primul microprocesor pe 64 de biți.

Încheiere

Chiar dacă *Intel* este cunoscută în primul rând datorită microprocesoarelor sale, acestea nu sunt singurele produse ale companiei.

Intel mai produce *chipset*-uri pentru plăci de bază, dispozitive de comunicații, memorii etc. Cu toate acestea, celebritatea firmei este datorată în primul rând microprocesorului, produs care a revoluționat industria calculatoarelor personale.

Carmen Șușman este redactor GInfo și poate fi contactată prin e-mail la adresa **schummy@email.ro**.

Istoria microprocesoarelor

Intel Pentium - 1993

Procesorul *Pentium* ușura manipularea datelor din lumea reală cum ar fi sunetele, fotografiile etc. Este primul procesor *Intel* care nu mai este identificat printr-un număr. Aceasta se datorează faptului că legile americane nu permit folosirea unui număr ca marcă înregistrată a unei companii.

Intel Pentium Pro - 1995

Acest procesor a fost proiectat pentru a satisface nevoile diferitelor aplicații, permițând creșterea vitezei aplicațiilor de proiectare asistată de calculator, a calculelor științifice etc. *Pentium Pro* conținea 5,5 milioane de tranzistori.

Intel Pentium II - 1997

Pentium II a fost proiectat în special pentru procesare video și audio și pentru grafică. Acest procesor facilita schimbul de informații *multimedia*, în special datorită setului de instrucțiuni *MMX* (acest set de instrucțiuni este prezent și la unele procesoare *Pentium* sau *Pentium Pro*).

Intel Pentium II Xeon - 1998

Acest nou membru al familiei de procesoare *Intel* a fost proiectat pentru a fi utilizat, în primul rând, pe *server*-e care rulează aplicații consumatoare de resurse. Puteau fi construite sisteme multiprocesor care să conțină patru sau opt astfel de procesoare.

Intel Celeron - 1999

Această versiune mai puțin performantă a procesorului *Pentium II* este destinată calculatoarele ieftine. *Celeron* oferă un raport preț/calitate excelent.

Intel Pentium III - 1999

Pentium III introduce un set de 70 de instrucțiuni noi care măresc performanțele aplicațiilor *3D*, de recunoaștere vocală etc. De asemenea, pot fi redate secvențe video de o calitate sporită, navigarea pe *Internet* este mai ușoară etc. Procesorul conține 9,5 milioane de tranzistori.

Intel Pentium III Xeon - 1999

Versiunea pentru *server*-e a procesoarelor din familia *Pentium III* oferă performanțe dosebite pentru realizarea de aplicații *e-Commerce* sau *multimedia*. La fel ca și *Pentium II Xeon*, poate fi folosit în sisteme multiprocesor.

Intel Pentium 4 - 2000

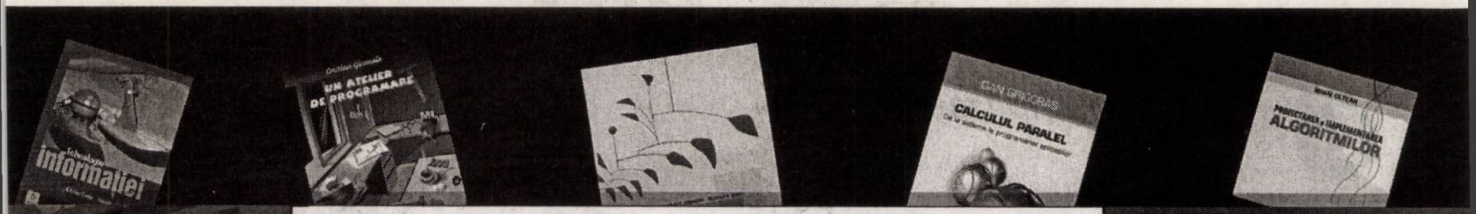
Chiar dacă denumirea firească ar fi fost *Pentium IV*, *Intel* a decis că e momentul pentru o schimbare, chiar dacă este una minoră. Folosind acest procesor, pot fi create filme de calitate profesională, pot fi vizualizate on-line secvențe video având o calitate comparabilă cu cea a televizoarelor, grafica *3D* este redată în timp real etc. Procesorul conține 42 de milioane de tranzistori.

Intel Xeon - 2000

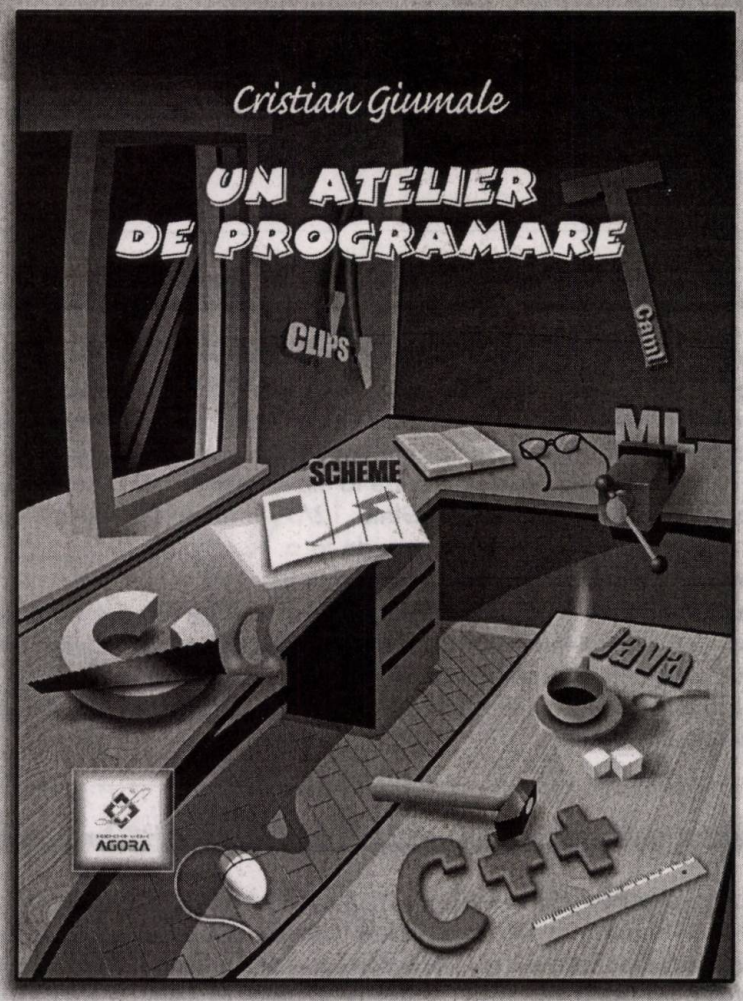
Versiunea pentru *server*-e aduce și ea o surpriză în denumire: procesorul se numește simplu *Xeon*. La fel ca și *Pentium III Xeon*, poate fi folosit în sisteme multiprocesor.

Intel Itanium - 2001

Acesta este primul produs din familia procesoarelor *Intel* pe 64 de biți. Este destinat numai *server*-elor și folosește o arhitectură cu totul nouă bazată pe tehnologia *EPIC* (*Explicitely Parallel Instruction Computing*).



COMPUTER LIBRIS AGORA



Cartea este o culegere de studii de caz, construite pe temelia problemelor concursurilor Southeastern European Regional Programming Contest din perioada 1995-1999, concursuri organizate sub egida Association for Computing Machinery.

Cartea se adresează atât începătorilor, cât și celor cu experiență în domeniul programării, în încercarea de a contura prin exemple imaginea unei lumi diverse și pasionante, dar pretențioase, aceea a înzestrării mașinii de calcul cu capacitatea de a rezolva corect și eficient.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

În numărul URMĂTOR

Numărul 12/3 (martie 2002) al Gazetei de Informatică va apărea în a doua jumătate a lunii următoare. Vă prezentăm pe scurt câteva dintre articolele pe care le veți putea citi...

Bursele Agora

Prima fază a concursului Bursele Agora s-a încheiat; în acest număr au fost publicate enunțurile problemelor de la ultima rundă. Începând cu următorul număr rubrica dedicată concursului nostru nu va mai conține noi enunțuri, ci doar rezolvările problemelor de la etapele precedente și diferite clasamente. În GInfo 12/3 veți putea găsi rezolvările celor trei probleme de la a treia rundă, numele și punctajele primilor 20 clasați la etapa a treia, precum și numele și punctajele celor care se află pe primele 50 de poziții ale clasamentului general. Dacă studiați clasamentele din acest număr și din numărul anterior, veți observa că nimeni nu a reușit să obțină toate cele 300 de puncte puse în joc la o rundă. În numărul următor veți putea afla care a fost copilăreasca greșeală care l-a împiedicat pe un concurent să devină primul care obține punctajul maxim la o anumită rundă.

IBM

Prezentarea companiilor care au avut o influență decisivă asupra dezvoltării tehnologiei informațiilor va continua în numărul următor. Veți putea citi istoria de peste o sută de ani a companiei IBM, una dintre cele mai mari corporații ale momentului.

Tehnici de criptare

Serialul dedicat criptării și decriptării mesajelor va continua în numărul următor. Veți găsi informații referitoare la diferite alte modalități de codificare a informațiilor pentru a spori securitatea transmiterii sau păstrării acestora.

Microsoft Visual Studio .NET

În cursul lunii februarie compania Microsoft a lansat o nouă versiune a pachetului de programe destinat programatorilor. În numărul următor veți putea citi o scurtă prezentare a suitei de programe care cuprinde, printre altele, mediile de programare Microsoft Visual C++ 7.0, Microsoft Visual Basic 7.0, Microsoft Visual J++ 7.0 etc. Noul pachet oferă tuturor programatorilor uneltele necesare dezvoltării de aplicații care să ruleze pe noua platformă Microsoft .NET.

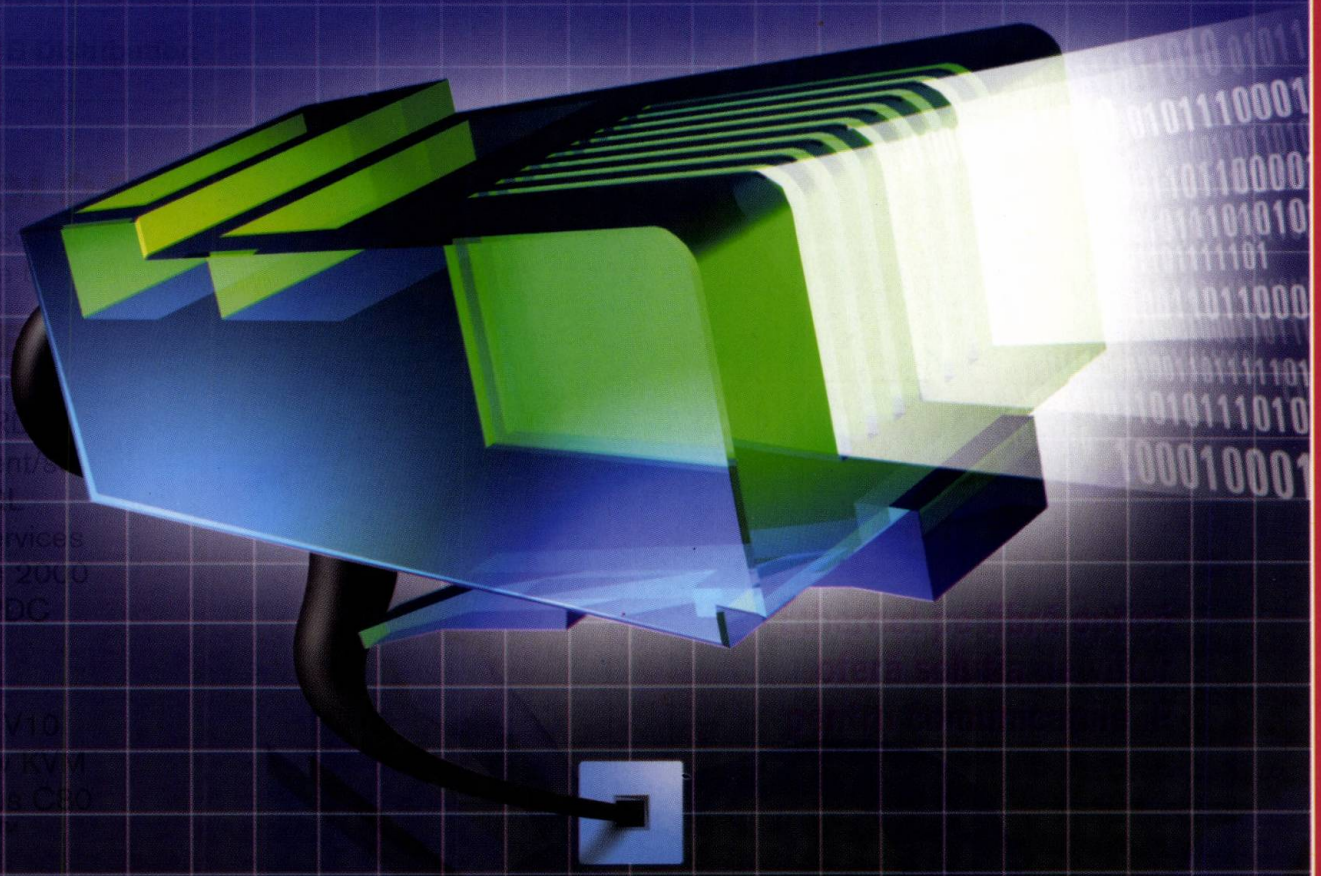
Heroes IV

Mult așteptata continuare a seriei Heroes of Might and Magic va apărea în cursul lunii martie. Veți putea citi o scurtă prezentare a noului joc care promite să fie cel puțin la fel de interesant ca și predecesorii săi. Cel de-al patrulea episod al seriei aduce o mulțime de noutăți importante care revoluționează modul în care va fi abordat acest joc de strategie.

Rețele de calculatoare

Este evident faptul că, în momentul în care se dorește crearea unei rețele de calculatoare, trebuie respectate anumite reguli. Acestea se referă, în primul rând, la mediul fizic (cabluri, conectori etc.) folosit. În numărul următor vor fi prezentate standardele care descriu regulile de cablare.

NETREPORT



Conectat la viitorul digital

AGORA MEDIA: Str. Tudor Vladimirescu nr. 63/9, Tg. Mureș, CP 230/1, 4300
Tel: 065-166 516, Fax: 065-166 290, e-mail: office@agora.ro

Agora MEDIA

www.netreport.ro

Publicarea acestei reviste este posibilă datorită sponsorizărilor acordate de următoarele firme care s-au decis să sprijine învățământul românesc de informatică

SPONSORI DE PLATINĂ

COMPAQ

SPONSORI DE AUR

FORTE COMPUTERS
Flexibility. Made in Romania

ASTRAL TELECOM

SPONSORI DE ARGINT

sdt
IT SOLUTIONS & SERVICES

hp
invent

ERICSSON

Magna

M.G.T.
EDUCATIONAL

a&C INTERNATIONAL S.A.
the CAD-CAMpany
autodesk
authorized distributor

ALCATEL

*Rugăm companiile care doresc să se alăture acestui efort să ne contacteze prin e-mail la adresa office@ginfo.ro sau la numărul de telefon: 065-166.516

Martie 2002
Vol. 12/3

Lei 20.000

gazeta de informatica

Revista viitorilor specialiști în tehnologia informației

din sumarul numărului:

Actual

Procesorul Mobile Intel© Pentium© 4

Profil IBM

Despre comunicare • Dispozitive de calcul • L-forme
Tehnici de criptare • Cablarea edificiilor • Probleme
Concursul interjudețean Grigore C. Moisil

GHIDUL TĂU ÎN LUMEA TEHNOLOGIILOR

PC MAGAZINE ROMÂNIA

de reținut!

CONCURS DE FOTOGRAFIE DIGITALĂ

HP România –
PC Magazine România

1 martie - 24 mai 2002

www.hp.com.ro, www.pcmagazine.ro

Misiunea noastră este de a testa și evalua produse și servicii din domeniul tehnologiei informației și comunicațiilor și de a prezenta rezultatele cu onestitate și obiectivitate

AGORA MEDIA Str. C-tin Rădulescu Motru nr. 13, etaj 4, apt. 42, sect. 4 (CP 94/49), București
Tel: 01-3309282, Fax: 01-3309285

www.pcmagazine.ro

Agora MEDIA

Vol. 12, Nr. 3 /2002

gazeta de informatică

Redacția:

Director științific:
prof. dr. *Horia Georgescu*
Redactor șef:
lect. univ. *Clara Ionescu*
Redactor șef adjunct:
Mihai Scorțaru
Redactori:
prof. *Eugen Ionescu*
Carmen Șușman
Claudiu Soroiu
Au mai colaborat:
Laviniu Bădulescu (Craiova)
Adrian Csibi (Cluj)
Lucian Ilea (Cluj)
Mihai Stroe (București)
Tehnoredactare:
Claudiu Soroiu
Coperta și grafica:
Andrei Pașa
Adresa pentru corespondență:
Computer Libris Agora,
Universității 8/8,
3400, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 064/192.422
065/166.516
Fax: 065/166.290
http://www.ginfo.ro
ISSN: 1222-7129

Tiparul: *Master Druck* Târgu Mureș
tel: 065/162.359

Editor: Agora Media SRL
Agora Media SRL, editor al publicațiilor "*eWeek România*", "*PC Magazine România*", "*NetReport*" și "*Gazeta de Informatică*", este membru al Biroului Român pentru Auditarea Tirajelor.
str. T. Vladimirescu nr. 63/9
4300 Târgu-Mureș
Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director tehnic:
Szabó László
Vânzare reclamă:
Mihai Bucuroiu
Bogdan Manu
Denis Marioara
Marius Comărniceanu
Reprezentanța București:
Director: *Dana Pastramă*
Tel: 01/330.92.81
Fax: 01/330.92.85

sumar

actual.. PAG. 5

Produsele lunii
Mobile Pentium 4
Visual Studio .NET
Heroes of Might and Magic IV
SOFTNEWS

probleme.. PAG. 9

Concursul "Grigore Mosil"
~Turda, jud. Cluj~

Concursul "Info-Oltenia"
~Târgu-Jiu, jud. Gorj~

Concursul "LInfo@SV"
~Suceava~

concurs.. PAG. 23

Rezolvările problemelor celei de-a treia runde

Clasamentul general

focus.. PAG. 28

L-câmpuri

Lucian ILEA
Andrei CSIBI

~problemă rezolvată~

babel.. PAG. 29

Despre concursuri

Mihai STROE

~psihologia concursurilor~
~de programare~

profil.. PAG. 34

Dispozitive MECANICE de calcul

Laviniu-Aurelian BĂDULESCU

International Business Machines

Carmen ȘUȘMAN

serial.. PAG. 40

Tehnici de CRIPTARE

Claudiu SOROIU

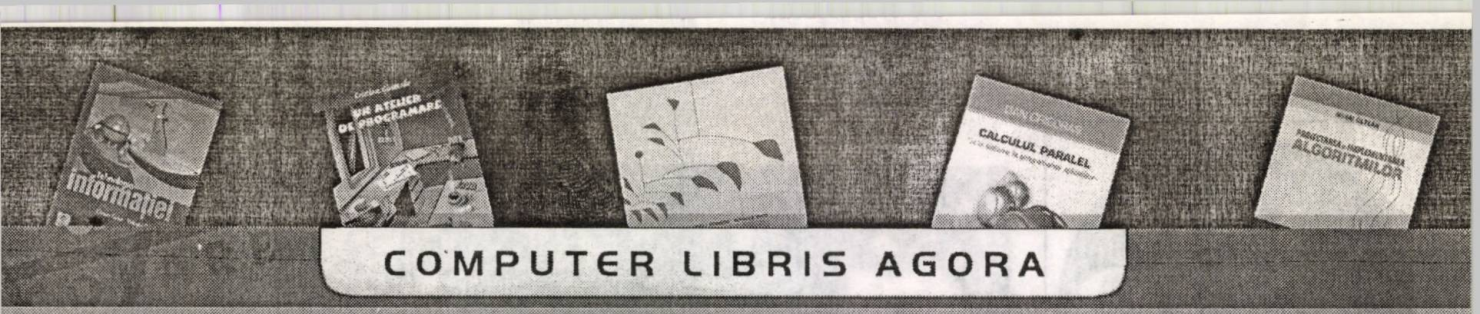
~al doilea episod~

tehno.. PAG. 43

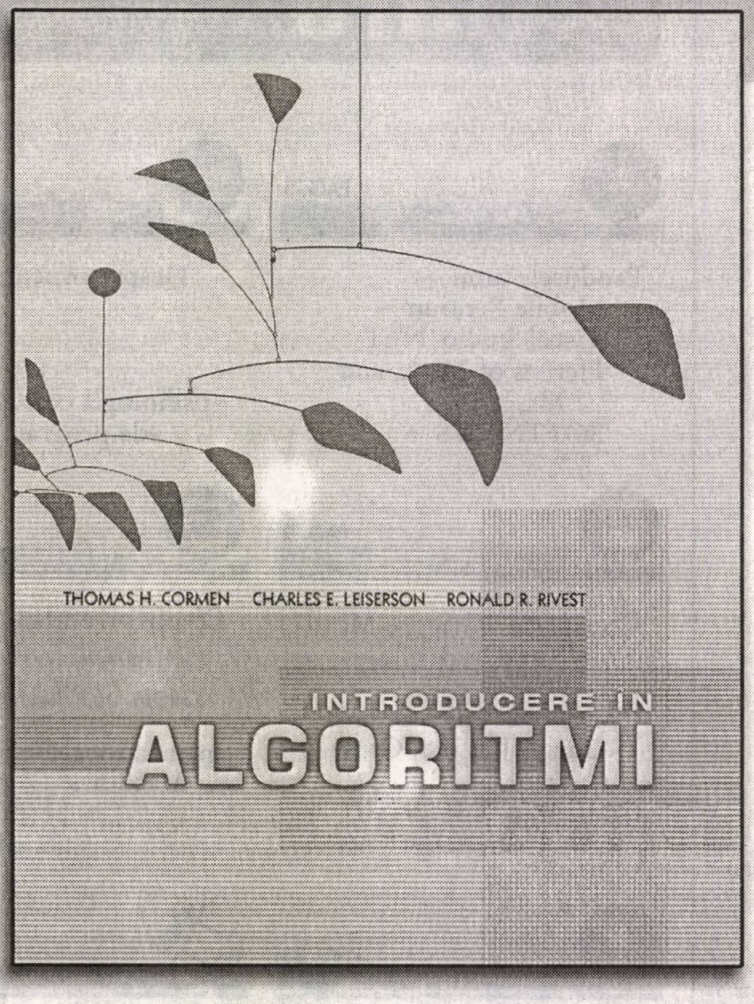
CABLAREA edificiilor

Mihai SCORȚARU

~Rețele de calculatoare~



COMPUTER LIBRIS AGORA



Cartea se adresează, în primul rând, celor care studiază la nivel elementar sau academic algoritmi și structuri de date. Deoarece tratează aspecte teoretice și practice, legate de proiectarea algoritmilor, este la fel de binevenită elevilor, studenților, profesorilor și autodidacților.
Introducere în algoritmi acoperă atât metodele clasice cât și metodele moderne ca analiza amortizată și algoritmii paraleli.
Prezentarea matematică, riguroasă, este atent detaliată în așa fel încât să fie accesibilă tuturor cititorilor.
Capitolele încep cu elemente de bază, avansând spre subiecte mai complexe.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

Speranțe...

Despre COMUNICARE

A venit *primăvara*! Cel puțin din punct de vedere calendaristic, dar parcă și în sufletele noastre se simte o adiere de prospețime, de reavăn, de nou, de dorința de a face ceva.

Cred că această stare, aceste porniri s-au manifestat în multitudinea de concursuri interjudețene de informatică care au avut loc în ultima vreme. Le voi aminti doar pe cele despre care am aflat, respectiv cele de la *Turda*, de la *Târgu-Jiu* și de la *Suceava*.

Organizatorii, profesori de informatică, simbolic vorbind, sunt cei care *seamănă* primăvara pentru a avea ce *ara* tot anul. Este o acțiune demnă de laudă care readuce în atenție munca de creator a dascălului.

După părerea mea aceste acțiuni au o mare importanță, nu numai prin faptul că oferă elevului doritor, pasionat de informatică prilejul de a se măsura cu el însuși și cu alții, dar mai ales din cauza că ele ***au loc*** (mai *seamănă* cineva și în România).

De aceea, nu voi pomeni de participanți, de subiecte, de complexitatea lor, de numărul de teste, de punctaje obținute, de clasamente și de câștigători. De altfel, astfel de amănunte puteți găsi câteva pagini mai încolo. La aceste acțiuni câștigători sunt cu toții atât profesori cât și elevi.

Contactul, schimbul de idei între participanți (elevi, studenți și profesori), este mai important decât rezultatul tehnic, aceasta și datorită faptului că un astfel de concurs nu are încărcătura competițională, emoțională a unei olimpiade naționale și de aceea, în opinia mea, este mult mai benefic pentru toți participanții.

Aș dori să remarc faptul că și la aceste concursuri participă studenți, foști olimpici care, alături de profesori, sunt sufletul acestor manifestări. Lăudabilă este dorința de a da, de a împărtăși cunoștințe și experiență într-o lume, unde egoismul și invidia câștigă tot mai mult teren.

M-am întristat, când redacția a fost acuzată că popularizează numai anumite concursuri și manifestări legate de informatică, neglijând altele pe criterii subiective (!?). Adevărul este că revista publică articole despre aceste manifestări în măsura în care află despre ele. De aceea, vă rugăm stimați colegi organizatori de concursuri de informatică, anunțați-ne despre desfășurarea unor astfel de evenimente și... certați-ne dacă v-am neglijat.

Remarc încă o dată că cele mai multe necazuri pleacă de la o problemă de comunicare sau lipsă de comunicare. Priviți un pic retrospectiv și amintiți-vă de câte ori s-a întâmplat să nu fiți înțeleși sau să fiți înțeleși greșit.

În această situație se poate ajunge din cauză că nu ați comunicat *ceva*, considerând că acel "ceva" este evident. Greșeala constă în presupunerea de evidență care implică pentru interlocutor un anumit grad de cunoaștere care nu poate fi măsurat, ci doar estimat.

Alte neînțelegeri sunt generate de ambiguitatea limbajului uman, mai precis de cunoașterea insuficient de profundă a limbajului fie de cel ce comunică, fie de cel căruia i se comunică, sau de interpretări personale ale unor noțiuni. Există o limitare a limbajului care, ca invenție a omului, este în continuă evoluție, suferă îmbogățiri, nuanțări, mai ales în ultimele decenii. În primele două cazuri, care sunt și cele mai frecvente, remediul ar fi unul sau mai multe exemple bine alese, astfel încât să se elimine orice ambiguitate care ar putea fi generată de una dintre situațiile descrise mai sus.

Revenind la concursurile noastre, constatăm că și aici ne confruntăm cu astfel de probleme de comunicare: elevii nu înțeleg enunțurile problemelor așa cum ar dori autorii și, de cele mai multe ori, "vina" nu este a elevilor.

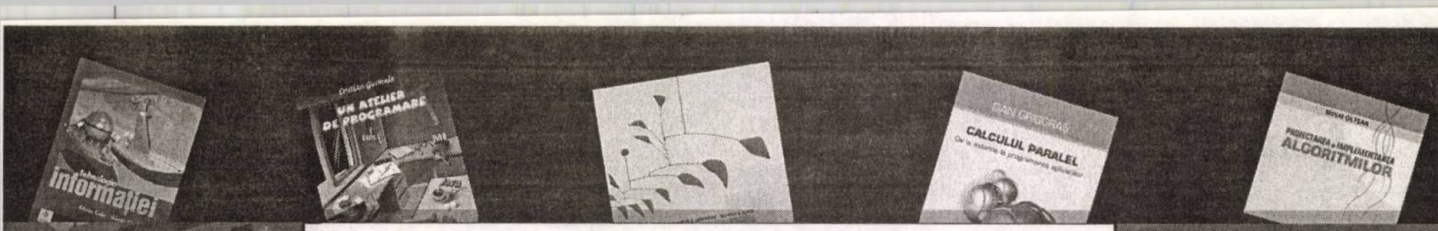
Un *enunț* care poate fi înțeles în mai multe feluri poate avea "calitatea" de a genera mai multe *probleme* (poate unele interesante), dar în nici un caz nu poate fi enunțul unei probleme de concurs de informatică. Vă rugăm, dragi autori de probleme de concurs, enunțați ***probleme*** și nu ghicitori de tipul "*cine ghicește ce a vrut autorul*" sau *teme de cercetare*.

Nu vă feriți de exemple sugestive, chiar dacă unii susțin că nu dau astfel de exemple pentru că, dacă le-ar da, ar sugera rezolvarea problemei. Dacă dând un exemplu, considerați că ați *rezolvat* problema, ea de fapt nu este o problemă ci o *situație*, eventual una de calcul, și în acest caz va trebui să căutați *problema* adevărată, dacă ea există și să o formulați, pornind chiar de la exemplele pe care le-ați evitat.

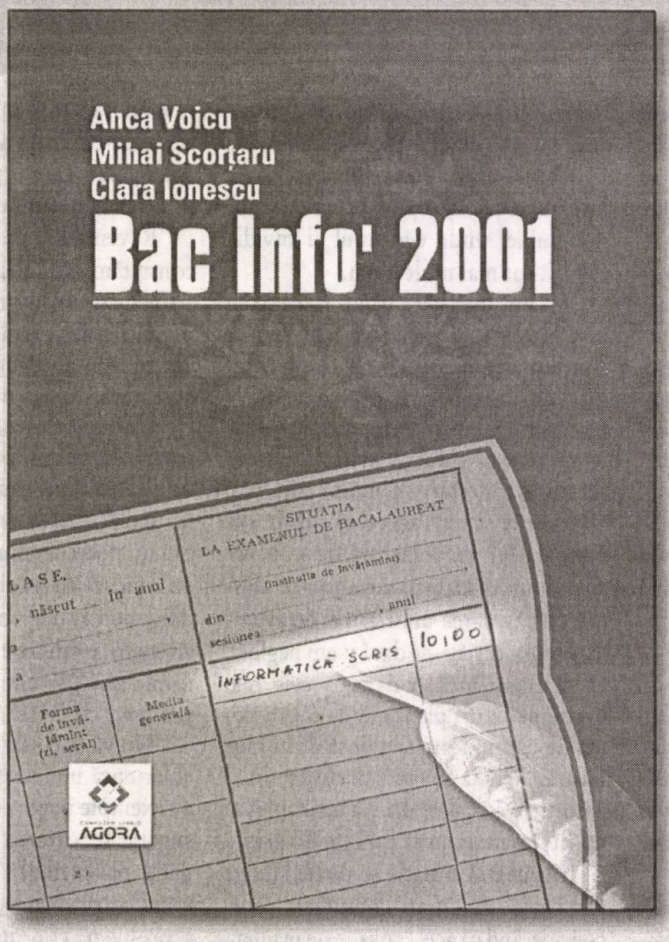
Competițiile, concursurile reușite, au avut întotdeauna la bază comunicarea, colaborarea, înțelegerea partenerului fie el elev, student sau profesor, rezolvitor sau propunător de probleme.

Dacă astfel de concursuri s-ar desfășura în cât mai multe locuri, poate că primăvara ar fi mai lungă și recolta ar fi mai bună, mai dătătoare de speranțe.

Eugen Ionescu



COMPUTER LIBRIS AGORA



Bac Info' 2001 conține 20 de variante de subiecte pentru Bacalaureat. Fiecare variantă conține câte patru subiecte. Primul dintre ele conține trei subpuncte care testează capacitățile candidaților de a recunoaște valoarea de adevăr a unei propoziții, de a determina efectul unui anumit algoritm și de a găsi o anumită greșeală într-un algoritm. Al doilea subiect conține doua subpuncte; primul testează deprinderea candidaților de a folosi limbajul pseudocod, iar al doilea cea de a folosi un limbaj de programare (Pascal sauC++). Al treilea subiect este o problemă pentru rezolvarea căreia este necesară folosirea unor structuri dinamice de date, iar al patrulea este o problemă pentru a cărei rezolvare trebuie folosită metoda backtracking.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

Mobile Pentium 4

Compania Intel a lansat la începutul acestei luni un nou procesor destinat calculatoarelor portabile: Mobile Pentium 4. Acesta este cel mai performant produs de acest tip disponibil în acest moment pe piață, motiv pentru care a devenit produsul lunii martie.

Introducere

Pe data de 4 martie 2002 a fost lansat procesorul *Mobile Pentium 4*, cel mai puternic procesor produs de compania *Intel*, destinat calculatoarelor portabile. Câteva companii, printre care *Compaq Computer*, *Dell Computer* și *Toshiba* au profitat de acest lucru și au pus în vânzare echipamente care utilizează noul procesor.

În aceeași zi, compania *Intel* a anunțat două noi *chipset*-uri destinate calculatoarelor portabile: *845MP* și *845MZ*. Primul dintre ele este prezent pe piață, iar cel de-al doilea va fi disponibil spre sfârșitul acestui an.

Date tehnice

Procesorul *Mobile Pentium 4* este disponibil în două variante care funcționează la frecvențe de 1,6 GHz, respectiv 1,7 GHz.

Folosind tehnologia *SpeedStep*, dezvoltată de compania *Intel*, noul procesor poate să funcționeze la tensiuni scăzute, fapt ce are ca efect creșterea duratei de viață a bateriei calculatorului portabil.

Această tehnologie permite procesorului să comute automat între două moduri de lucru: performanță maximă (*Maximum Performance mode*) și optimizare pentru baterie (*Battery Optimized Performance mode*), în funcție de cantitatea de date pe care procesorul trebuie să le prelucreze la un moment dat.

Trecerea de la un mod de lucru la celălalt se realizează prin modificarea frecvenței magistralei, a tensiunii de funcționare și a vitezei de lucru a procesorului fără a fi necesară reinițializarea sistemul.

Noile procesoare au fost construite folosind tehnologia de 0,13 μm și includ seturile de instrucțiuni *MMX*, *SSE* și *SSE2*.

La fel ca la actualele procesoare *Intel Pentium 4* pentru sistemele *desktop*, frecvența magistralei sistem este de 400 MHz.

Mobile Pentium 4 conține două unități aritmetico-logice (*ALU*) care, la fel ca și în cazul celorlalte procesoare din familia *Pentium* 4, lucrează la o frecvență dublă față de cea a procesorului.

Capacitatea memoriei *cache Level 2* este de 512 KB.

Performanțe

Cele două procesoare din familia *Mobile Pentium 4* sunt capabile să acceseze memorii cu dimensiuni de până la 4 GB.

Datorită magistralei de mare viteză, dimensiunea datelor care pot fi transmise între placa de bază și procesor într-o secundă, poate atinge 3,2 GB.

Unitatea pentru operații în virgulă flotantă folosește regiștri pe 128 de biți și a fost introdus un nou registru pentru transferul datelor, lucru care duce la o creștere semnificativă a performanțelor aplicațiilor multimedia.

Procesoarele *Mobile Pentium 4* au fost optimizate pentru conectivitate în rețele *wireless*. Datorită acestui fapt, se pot transfera date într-un mod eficient (fără multe erori), chiar și la viteze mari de transmisie.

Intel a efectuat teste pe două sisteme care aveau o configurație aproape identică, singura diferență fiind dată de frecvența procesorului. Primul sistem a fost echipat cu un *Mobile Pentium 4* la 1,6 GHz, iar celălalt cu un *Mobile Pentium 4* la 1,7 GHz. Aceste teste indică o diferență foarte mică, practic nesemnificativă pentru calculatoarele portabile (pentru care viteza procesorului are un rol mult mai mic decât în cazul *desktop*-urilor), a performanțelor celor două procesoare.

Încheiere

Datorită noilor caracteristici amintite anterior și a faptului că noile procesoare nu au fost lansate de foarte mult timp, prețul lor este destul de mare.

Acesta se ridică la 508$ pentru un procesor *Mobile Pentium 4* cu frecvența de 1,7 GHz și 401$ pentru un procesor *Mobile Pentium 4* la 1,6 GHz. În momentul lansării, prețul unui *chipset 845MP* era de 43$.

Chiar dacă *Mobile Pentium 4* este cel mai rapid procesor destinat calculatoarelor portabile, unii analiști afirmă că *cel mai rapid procesor disponibil azi consumă cea mai multă energie* și că, *în mediul calculatoarelor portabile, un procesor mai rapid nu înseamnă neapărat un procesor mai bun.*

Visual Studio .NET

Microsoft Corporation a lansat în cursul lunii februarie o nouă versiune a pachetului său de programe destinat dezvoltării de aplicații. Noul produs se numește Visual Studio .NET și introduce o serie de inovații datorită cărora acesta a devenit cel mai important produs de acest tip lansat pe piață în ultima perioadă.

Introducere

Microsoft a lansat în luna februarie a acestui an un nou pachet de programe destinate dezvoltării de aplicații, și anume *Visual Studio .NET*.

Acest pachetul conține, spre deosebire de versiunile anterioare, un singur mediu de programare destinat dezvoltării de aplicații în mai multe limbaje de programare, în funcție de preferințele celor ce utilizează acest produs.

Noul pachet oferit de *Microsoft* programatorilor de nivel mediu și avansat este disponibil în trei variante diferite: *Visual Studio .NET Enterprise Architect* (*VSEA*), *Visual Studio .NET Enterprise Developer* (*VSED*) și *Visual Studio .NET Professional*.

Descriere

Prima variantă, *Visual Studio .NET Enterprise Developer*, oferă unelte avansate destinate realizării de aplicații de generația următoare pentru sisteme de operare din familia *Windows* și pentru *Internet*. Acest pachet introduce facilități noi față de versiunile anterioare ale produsului *Visual Studio* pentru dezvoltarea de aplicații critice care au ca țintă orice dispozitiv de conexiune la *Internet* și integrarea cu orice platforme.

De asemenea, pot fi create aplicații *Web*, bazate pe tehnologia *XML*, iar programarea în echipe care folosesc acest produs este mult mai eficientă deoarece un membru poate să acceseze în siguranță codul sursă creat de alți membri.

Visual Studio .NET Enterprise Architect se deosebește de pachetul anterior prin faptul că programatorii pot defini foarte ușor, folosind unelte *XML*, funcționalitatea și arhitectura aplicațiilor sau serviciilor *Web*. De asemenea, se pot crea scheme *UML* (*Unified Modeling Language*) pentru aplicații, lucru util deoarece arhitectura și funcționalitatea unei aplicații sunt mult mai ușor de înțeles.

Folosind modele *UML* pentru aplicații, se reduce foarte mult timpul afectat pentru crearea unei aplicații sau a unui modul, în special pentru limbajele de programare *Microsoft Visual C++ .NET*, *Microsoft Visual Basic .NET* și *Microsoft Visual C# .NET*. Aceste modele pot fi create folosind aplicația *Microsoft Visio*.

De asemenea, programatorii pot să realizeze mai ușor documentații pentru aplicații scrise în unul din limbajele enumerate mai sus, creând scheme *UML* prin folosirea *ingineriei inverse*.

Pachetul *Visual Studio .NET Enterprise Architect* oferă suport integral pentru modelarea aplicațiilor care folosesc suportul integrat pentru baze de date.

Cea de-a treia variantă în care este disponibil acest pachet poartă numele *Microsoft Visual Studio .NET Profesional* și oferă programatorilor un mediu practic pentru realizarea de aplicații pentru sistemul de operare *Windows* și pentru *Internet*.

Programatorii se pot folosi de uneltele *XML* și *Microsoft ADO.NET* în realizarea de aplicații performante pentru lucrul cu baze de date.

Una dintre facilitățile importante ale utilizării mediului de programare *Microsoft Visual Studio .NET* este depanatorul integrat de programe, care include acum suport pentru depanarea de aplicații scrise în mai multe limbaje. De exemplu, se pot depana aplicații scrise în *Visual Basic .NET* și *Visual C# .NET* în aceeași sesiune de lucru.

Folosind acest pachet, programatorii pot crea foarte repede aplicații care au ca țintă integrarea dispozitivelor *Internet* cu orice platformă.

Concluzii

Folosirea unui singur mediu de programare pentru toate limbajele de programare, în consecință aceleași unelte și același depanator de programe, reduc foarte mult timpul necesar pentru realizarea unei aplicații de către o echipă, aplicație care poate folosi module ale căror surse au fost scrise în diferite limbaje.

Se pare că produsul *Microsoft Visual Studio .NET* este unul dintre cele mai puternice medii de programare la ora actuală. Cu ajutorul acestuia se pot realiza aplicații destinate celor mai diverse domenii.

Heroes of Might & Magic IV

La sfârșitul lunii martie este lansat al patrulea joc al seriei Heroes of Might and Magic. Chiar dacă principiul de bază este același, noutățile introduse de producători fac ca acest joc să devină unul dintre cele mai bine realizate jocuri de strategie bazate pe ture.

Istorie

Seria *Heroes of Might and Magic* s-a născut la mijlocul deceniului trecut. Primul joc al acesteia a impresionat prin calitatea grafică deosebită și prin satisfacțiile pe care jucătorii le aveau după fiecare succes important. Existau patru tipuri de castele, patru tipuri de eroi și în jur de 25 de tipuri de creaturi.

Cel care a impus această serie a fost *Heroes of Might and Magic* II, versiune care a primit titlul de cel mai bun joc al anului 1996. Numărul tipurilor de castele a crescut la șase, existau douăsprezece tipuri de eroi care puteau controla peste 50 de tipuri de creaturi.

Următoarea versiune a jocului nu a adus modificări deosebite din punct de vedere al conceptului de joc. Grafica a fost mult îmbunătățită, dar principiile au rămas aceleași.

Au existat trei variante ale acestui joc: *Restoration of Erathia*, *Armaggedon's Blade* și *Shadow of Death*. În prima variantă jucătorul avea la dispoziție opt tipuri de castele, șaisprezece tipuri de eroi și peste 100 de creaturi. *Armaggedon's Blade* a introdus un nou tip de castel, împreună cu două noi tipuri de eroi și peste 20 de creaturi noi.

Heroes of Might and Magic III a fost completat de seria *Heroes Chronicles*, opt jocuri care erau, de fapt, simple campanii ale jocului *Heroes* III.

Între timp a apărut și o versiune neoficială numită *In the Wake of Gods* care a fost dezvoltată de un grup de entuziaști și care realizează trecerea de la *Heroes* III la *Heroes* IV, introducând o mulțime de posibilități noi pentru acest joc.

Firul epic

După ce un cataclism a lovit *Erathia*, *Emilia Nighthaven* a pierdut totul. Rămasă orfană în mijlocul unei lumi bizare, ea încearcă să-i organizeze pe supraviețuitorii înspăimântați pentru a se putea apăra de cei care vor să domine noua lume.

Între timp apare *Magnus*, regele nemuritor al vrăjitorilor, și îi cere să i se supună. Acesta are la dispoziție un dispozitiv prin care preia controlul asupra minților celor ce se apropie de el.

Cum ar putea *Emilia* să-și apere regatul împotriva unei asemenea puteri? În scurt timp, datorită influenței lui *Magnus*, cei mai buni prieteni îi devin dușmani. Din nou, ea este cu adevărat singură. O rază de speranță apare atunci când unul dintre inamici trece de cealaltă parte a baricadei, aliindu-se cu *Emilia*...

Prezentare

Heroes of Might and Magic IV folosește un nou motor grafic, cu ajutorul căruia se pot obține efecte interesante.

Jocul introduce un nou sistem de magie: cele patru școli tradiționale (*aer*, *apă*, *foc* și *pământ*) au fost înlocuite cu cinci școli noi: *natură*, *viață*, *ordine*, *moarte* și *haos*. Fiecare dintre cele cinci școli de magie este aliată cu alte două școli: dacă ne imaginăm că cele cinci școli se află în colțurile unei stele, atunci fiecare școală este aliată cu cele două școli vecine. De exemplu, școala *vieții* este aliată cu cea a *naturii* și cu cea a *ordinii*.

Numărul tipurilor de orașe a fost redus la șase: *Preserve*, *Haven*, *Academy*, *Necropolis*, *Asylum* și *Stronghold*. Primele cinci corespund, în această ordine, celor cinci școli de magie. Cel de-al șaselea oraș nu are capacități magice.

Cele mai multe noutăți introduse de *Heroes* IV apar în cazul eroilor. Acum, aceștia vor participa efectiv la lupte, putând ataca creaturile inamice. Până la versiunea anterioară a jocului, eroii aveau un rol pasiv, de a transporta armatele pe hartă și de a lansa vrăji în timpul luptelor. O altă noutate este faptul că acum există posibilitatea ca mai mulți eroi să însoțească un grup de creaturi, sau ca un grup de creaturi să nu fie însoțite de nici un erou. În total, există 48 de tipuri de eroi.

Numărul monștrilor a fost redus și el la 66, fiecare dintre aceștia având câte o caracteristică proprie.

Încheiere

Noua lume *Heroes* așteaptă să fie explorată. De-a lungul celor șase campanii veți afla totul despre acest joc. În numărul următor veți putea citi un *review* complet al jocului, prima prezentare de acest tip publicată în *GInfo*.

SOFTNEWS

Mulți se întreabă de ce site-urile românești nu oferă la fel de multe informații ca cele din afara granițelor. Probabil răspunsul este legat de tranziție, de dificultățile economice etc. Totuși, primii pași spre site-urile de calitate a fost făcut. În domeniul software, a apărut un site românesc care poate rivaliza cu orice site de acest tip din lume. Acesta este foarte bine realizat și oferă o mulțime de informații în domeniu; www.softnews.ro a devenit site-ul lunii martie.

Prezentare

La **www.softnews.ro** veți găsi un *site* dedicat unui domeniu foarte vast: produsele *software*.

Creatorii *site*-ului vă pun la dispoziție ultimele informații în domeniu; de multe ori acestea apar înainte ca ele să fie publicate pe *site*-urile consacrate care se adresează aceluiași segment de utilizatori.

Probabil una dintre cele mai importante secțiuni ale *site*-ului este cea dedicată descărcării diferitelor programe. Oficial, *site*-ul duce o politică antipiraterie susținută dar, uneori, au apărut unele mici scăpări, pe *site* fiind disponibile programe pentru care nu existau drepturi de distribuție. Se pare că această situație s-a remediat, iar acum toate produsele *software* care sunt descărcate de pe acest *site* pot fi folosite în mod legal.

De multe ori, unele programe sunt disponibile în premieră pe acest *site*. Au fost cazuri în care un anumit produs a fost disponibil pentru *download* la **www.softnews.ro** inainte de a apărea pe *site*-ul producătorului.

Totuși, o parte dintre *link*-urile spre programele care ar trebui să poată fi descărcate nu sunt funcționale.

O secțiune deosebit de importantă a *site*-ului este forumul de discuții. Aici puteți să cereți părerea partenerilor de discuție referitor la orice subiect din domeniul *software* și nu numai. Majoritatea soluțiilor la problemele cu care vă confruntați le veți putea găsi tot aici.

Promovare

Totul a început când un grup de tineri au anunțat pe un forum de discuții că în curând va fi lansat "*cel mai tare site de soft din România*".

Creatorii *site*-ului au început să-l promoveze cu mult înainte ca acesta să fie funcțional. Acest fapt le-a adus foarte multe critici, dar probabil scopul a fost atins: foarte mulți au aflat de intenția unui grup de pasionați de a crea un *site* dedicat domeniului *software* și, când *site*-ul a fost în sfârșit lansat, acesta era destul de bine cunoscut.

În scurt timp, numărul celor care contribuie la realizarea *site*-ului a crescut, foarte mulți pasionați fiind atrași de idee.

Din păcate, politica de promovare a *site*-ului a devenit foarte agresivă. Din nou scopul a fost probabil atins, dar creatorii și-au atras antipatia multora. Mai mult, ei acceptau foarte greu criticile și aceasta a contribuit și mai mult la "izolarea" lor.

Cu toate acestea, *site*-ul a devenit în scurt timp unul dintre cele mai vizitate *site*-uri românești deoarece este foarte bine realizat, oferă o mulțime de informații utile, care pot fi găsite foarte ușor.

Domeniile principale asupra cărora se concentrează acest *site* sunt ***utilitare***, ***multimedia***, ***Internet*** și ***aplicații sistem***.

Încheiere

Acest *site* este unul dintre cele mai utile *site*-uri românești. Existența sa demonstrează faptul că primii care pun în practică o anumită idee vor obține rezultate spectaculoase, cel puțin atâta timp cât nu va exista concurență.

De asemenea, se poate observa cum o politică agresivă de promovare poate avea succes, chiar dacă prin astfel de procedee pot apărea conflicte cu diverse persoane. Totuși, *site*-ul s-ar fi putut impune și prin mijloace "tradiționale", probabil într-un timp puțin mai lung.

Vă recomandăm să vizitați acest *site* dacă sunteți interesați de domeniul *software* sau dacă aveți nevoie de diferite aplicații.

De asemenea, dacă frecventați diferite forumuri de discuții, va trebui să vă avertizăm că este mai bine să ignorați excesele care se fac pentru promovarea acestui *site*.

Site-ul este, cu siguranță, "*cel mai tare site de soft din România*". Totuși, creatorii săi ar trebui să realizeze că nu toți utilizatorii vor avea această părere și să accepte criticile acestora, mai ales dacă acestea sunt constructive.

Turda, județul Cluj

Concursul interjudețean GRIGORE C. MOISIL

În perioada 1-3 Martie 2002 a avut loc la Colegiul Tehnic din Turda cea de-a XVII-a ediție a concursului interjudețean de matematică și informatică "Grigore C. Moisil". Vă prezentăm în continuare enunțurile problemelor propuse spre rezolvare la secțiunea de informatică a acestui concurs.

P030201: Lacăt

(clasa a IX-a)

Seiful din biroul domnului director de la *Colegiul Tehnic Turda* are ca cifru un număr natural N. Cheia electronică folosită pentru deschiderea acestuia este un număr natural mai mic decât N. Directorul, fiind profesor de matematică, scrie pe o foaie toate numerele pe care le folosește, dar uită care dintre ele este cheia seifului. Cheia care deschide seiful are o reprezentare binară care, prin suprapunere peste o secvență din descompunerea binară a cifrului lacătului, modifică cifrul după următoarele reguli:

- 0 peste 1 nu modifică cifra;
- 1 peste 0 transformă cifra în 1;
- 1 peste 1 transformă cifra în 0;
- 0 peste 0 nu modifică cifra.

Lacătul se deschide dacă reprezentarea binară a cifrului lacătului, după suprapunere, conține doar cifre binare cu valoarea 1.

Ajutați conducerea liceului să caute printre numerele date cheile corecte, doarece toate premiile sunt închise în seif.

Date de intrare

Fișierul de intrare **SEIF.IN** conține pe prima linie cifrul seifului. Pe a doua linie se află numărul N al numerelor de pe foaie. Pe a treia linie se află cele N numere, separate printr-un singur spațiu.

Date de ieșire

Fișierului de ieșire **CUVINTE.OUT** va conține o singură linie pe care se vor afla cheile corecte, separate prin spații.

Restricții și precizări

- $1 \le N \le 100$;
- valorile numerelor și ale cifrului sunt cuprinse între 1 și 2000000000.

Exemplu

SEIF.IN

```
43
7
3 7 5 17 8 35 10
```

SEIF.OUT

```
5 10
```

Explicație

Reprezentarea binară a valorii 43 este 101011, reprezentarea binară a valorii 5 este 101, iar cea a valorii 10 este 1010. Numerele se suprapun astfel:

```
101011      101011
 101         1010
```

Timp de execuție: 1 secundă/test

P030202: Cuvinte

(clasa a IX-a)

Se consideră N cuvinte formate doar din majuscule. Acestea sunt despărțite în silabe cu ajuroul caracterului '-' (*minus*). Se spune că două cuvinte *rimează* dacă și numai dacă ultima lor silabă coincide.

Sarcina voastră este de a scrie un program care împarte cele N cuvinte în grupe, astfel încât oricare două cuvinte aflate în aceeași grupă să rimeze și oricare două cuvinte aflate în grupe diferite să nu rimeze.

După determinarea grupelor, se cere să se caute o frază formată din cât mai multe cuvinte, astfel încât cuvintele simetrice față de mijlocul frazei să rimeze (primul cuvânt va rima cu ultimul, al doilea cu penultimul etc.).

Date de intrare

Fișierul de intrare **CUVINTE.IN** conține pe prima linie numărul N al cuvintelor.

Pe fiecare dintre următoarele N linii se va afla câte un cuvânt.

Date de ieșire

Pe prima linie a fișierului de ieșire **CUVINTE.OUT** se va afla numărul K al grupelor formate. Pe fiecare dintre următoarele K linii se vor afla cuvintele dintr-o anumită grupă. Pe ultima linie a fișierului se va afla fraza determinată. Cuvintele de pe o linie vor fi separate printr-un singur spațiu.

Restricții și precizări

- $1 \le N \le 100$;
- dacă există mai multe fraze, trebuie furnizată doar una dintre ele.

Exemplu

CUVINTE.IN

```
10
MI-SI-U-NE
MI-NU-NE
CA-FEA
LU-LEA
MAR-GEA
A-LO-CU-TI-U-NE
DI-MEN-SI-U-NE
DOI-LEA
CAZ
MA-CAZ
```

CUVINTE.OUT

```
5
MISIUNE MINUNE ALOCUTIUNE DIMENSIUNE
LULEA DOILEA
CAZ MACAZ
CAFEA
MARGEA
MISIUNE ALOCUTIUNE LULEA MACAZ MARGEA CAZ
                        DOILEA MINUNE DIMESIUNE
```

Timp de execuție: 1 secundă/test

P030203: Hoții

(clasa a X-a)

La o bancă s-a produs un jaf; experții au descoperit urme pe un covor pătrat de culoare deschisă. Ei au împărțit covorul într-o rețea de pătrate foarte densă, au ales un sistem de coordonate și au marcat pozițiile înnegrite de pe covor. Tot din cercetări a rezultat că nu există doi hoți ale căror urme să fie identice. O urmă este formată dintr-o porțiune compactă de poziții înnegrite. Două poziții înnegrite învecinate (pe orizontală, verticală sau diagonală) fac parte din aceeași urmă. Să se identifice numărul hoților și, pentru fiecare urmă, hoțul căruia îi aparține.

Două urme nu pot fi lipite sau suprapuse și se află integral pe covor. Două urme aparțin aceluiași hoț dacă pot fi suprapuse prin rotația și/sau oglindirea uneia dintre ele.

Date de intrare

Fișierul de intrare **IN.TXT** va conține pe prima linie dimensiunea N a rețelei (covorului). Pe următoarele linii se află câte două numere naturale, separate printr-un spațiu, reprezentând linia și coloana unde s-a găsit o porțiune dintr-o urmă.

Date de ieșire

Fișierul de ieșire **OUT.TXT** va conține pe prima linie un număr natural NR care va indica numărul hoților. Pe fiecare dintre următoarele N linii se vor afla N numere separate prin câte un spațiu. O poziție fără urme va fi codificată cu 0, iar una cu urme prin numărul de ordine al hoțului căruia îi aparține. Numerele de ordine ale hoților sunt alese arbitrar, dar sunt distincte și sunt cuprinse între 1 și NR.

Restricții și precizări

- $1 \le N \le 20$;
- există cel mult zece hoți.

Exemplu

IN.TXT	**OUT.TXT**
10	3
2 1	0 0 0 0 0 0 0 0 0 0
2 2	1 1 0 0 0 0 0 3 0 0
3 2	0 1 0 1 1 0 0 0 0 0
4 2	0 1 0 1 0 0 0 0 0 0
3 4	0 0 0 1 0 2 2 0 0 0
3 5	0 0 0 0 0 2 2 0 0 0
4 4	0 0 1 0 0 0 0 0 0 0
5 4	0 0 1 0 0 1 0 0 0 0
5 6	0 1 1 0 0 1 0 0 0 0
5 7	0 0 0 0 0 1 1 0 0 0
6 6	
6 7	
7 3	
8 3	
9 3	
9 2	
8 6	
9 6	
10 6	
10 7	
2 8	

Timp de execuție: 1 secundă/test

P030204: Cuvinte magice

(clasa a X-a)

Se dau n cuvinte, formate din litere mici ale alfabetului englez.

Să se formeze cuvintele de lungime maximă, folosind litere din aceste cuvinte, respectând următoarele reguli:

- a i-a literă a cuvântului este o literă de pe poziția i a unui cuvânt dat;
- literele fiecărui cuvânt sunt ordonate lexicografic.

Date de intrare
Fișierul de intrare `MAGIC.IN` conține cuvinte, câte unu pe fiecare linie.

Date de ieșire
Fișierul de ieșire `MAGIC.OUT` conține cuvintele magice, câte unul pe fiecere rând.

Restricții și precizări
- fișierul de intrare conține cel mult 20 de cuvinte;
- fișierul de ieșire va conține cel mult 1000 de cuvinte magice;
- fiecare cuvânt va conține cel mult 20 de litere.

Exemplu

`MAGIC.IN`	`MAGIC.OUT`
la	luv
revedere	ruv
turda	tuv

Timp de execuție: 5 secunde/test

P030205: Dreptunghiuri

(clasele a XI-a și a XII-a)

Se cunosc dimensiunile (lungimea și lățimea) a N dreptunghiuri. Să se determine, dacă este posibil, o aranjare a dreptunghiurilor astfel încât acestea să acopere în întregime un pătrat de latură LP. Trebuie folosite toate dreptunghiurile. Acestea pot fi rotite în plan, dar laturile lor trebuie să rămână paralele cu laturile pătratului. Dreptunghiurile nu se pot suprapune și nu pot ieși din pătrat.

Date de intrare
Fișierul de intrare `DR.IN` conține pe prima linie numărul N al dreptunghiurilor. Fiecare dintre următoarele N linii conține câte două numere care reprezintă lungimile laturilor unui dreptunghi. Pe ultima linie se află latura pătratului.

Date de ieșire
Fișierul de ieșire `DR.OUT` va conține o matrice cu LP linii și LP coloane. Elementele matricei au valori cuprinse între 1 și N, iar valorile de pe o linie vor fi separate prin câte un spațiu. Elementele matricei care corespund dreptunghiului cu numărul de ordine i vor avea valoarea i.

Dacă aranjarea nu este posibilă, fișierul de ieșire va conține o singură linie pe care se va afla șirul de caractere `IMPOSIBIL`.

Restricții și precizări
- $1 \le N \le 20$;
- $1 \le LP \le 50$;
- laturile dreptunghiurilor sunt cel mult egale cu latura pătratului;
- dacă există mai multe soluții se va determina doar una dintre ele.

Despre concurs...

Concursul interjudețean "Grigore C. Moisil" a devenit unul tradițional în nord-vestul țării. Este unul dintre puținele concursuri din România la care se întâlnesc matematicienii și informaticienii. Chiar dacă secțiunea dedicată matematicii este mai bine cunoscută, nivelul concursului de informatică este destul de ridicat.

La secțiunea dedicată informaticii de la această ediție a concursului au participat 25 de elevi din județele Bihor, Bistrița-Năsăud, Cluj, Maramureș, Satu Mare și Sălaj.

Este pentru prima dată când acest concurs se desfășoară la Turda. Domnul profesor Dumitru Mateescu, directorul Colegiului Tehnic din Turda, a declarat: ***"A fost un concurs de împrejurări pentru ca ediția din acest an să se desfășoare la noi în școală, concurs de împrejurări, zic eu, fericite. În primul rând, nici un liceu din Municipiul Cluj-Napoca nu putea să găzduiască aproape 300 de persoane, în al doilea rând, școala noastră dispune de toată logistica necesară unui concurs de asemenea anvergură."***

Premiile au fost destul de mari, totalizând peste 50 de milioane lei și cărți în valoare de aproximativ 30 de milioane.

Întrebat cum a reușit să obțină aceste fonduri, domnul director a spus: ***"Nu am apelat la sponsori. Am avut doar sprijinul unor instituții de stat: Consiliul Local Cluj-Napoca, Inspectoratul Școlar Județean Cluj și Consiliul Local Turda"***.

Cu toate acestea sponsorii au apărut în momentul festivității de premiere. Editura "Altip" din Alba Iulia a oferit premii în valoare de peste 20 de milioane de lei.

Gazeta de Informatică a fost reprezentată la acest concurs de redactorul-șef adjunct Mihai Scorțaru, precum și de redactorii Claudiu Soroiu și Carmen Șușman.

Exemple

`DR.IN`	`DR.OUT`
8	1 1 1 3 4 4 5
2 3	1 1 1 3 4 4 6
3 5	2 2 2 3 4 4 7
1 7	2 2 2 3 8 8 8
2 3	2 2 2 3 8 8 8
1 1	2 2 2 3 8 8 8
1 1	2 2 2 3 8 8 8
1 1	
4 3	
1	

DR.IN	DR.OUT
3 1 1 1 1 1 2 2	1 2 3 3

DR.IN	DR.OUT
4 2 2 1 2 1 2 1 2 3	IMPOSIBIL

Timp de execuție: 1 secundă/test

P030206: Atlantis

(clasele a XI-a și a XII-a)

După ce a reușit să treacă de poarta cetății *Atlantis*, *Iahim Uratrox*, marele explorator al *Atlantidei*, a descoperit o mulțime dintre vestigiile vechii civilizații. Singura clădire în care celebrul explorator nu a putut intra este palatul foștilor suverani atlanți.

Se presupune că în interiorul acestui palat se pot găsi documentele care să explice motivul scufundării miticului oraș. Din nefericire, intrarea în palat este protejată de un cifru.

Mecanismul este foarte simplu, el constând din doar două dispozitive de introducere a codului. Pe mecanism este gravat un număr format din cinci cifre.

Bineînțeles, savanții au reușit imediat să descopere algoritmul de funcționare al mecanismului. Cele două dispozitive corespund semnelor `'+'` și `'-'`.

Fiecare atingere a unui dispozitiv duce la adunarea sau scăderea unui număr. Acest număr variază în funcție de numărul de ordine al atingerii. La a i-a atingere, numărul care se adună sau se scade este i^2. Se pornește cu valoarea 0.

Așadar, la fiecare atingere i valoarea corespunzătoare mecanismului scade sau crește cu i^2. În final, pentru a deschide poarta, trebuie să se obțină valoarea gravată pe mecanism.

Date de intrare

Numele fișierului de intrare este **ATLANTIS.IN**. Acesta conține o singură linie pe care se află cele cinci cifre ale numărului gravat pe mecanismul cifrului, neseparate prin spații. Există posibilitatea ca prima, primele două, primele trei sau primele patru cifre să fie 0. Totuși, cel puțin una dintre cele cinci cifre este diferită de zero.

Date de ieșire

Numele fișierului de ieșire este **ATLANTIS.OUT**. Acesta conține o singură linie pe care se vor afișa semnele corespunzătoare atingerilor mecanismului.

Al i-lea semn de pe această linie va corespunde celei de-a i-a atingeri. Dacă prin această atingere se adună valoarea i^2, atunci semnul va fi `'+'`. Dacă atingerea duce la scăderea valorii i^2, atunci semnul va fi `'-'`.

Restricții și precizări

- Valoarea numărului inscripționat pe mecanismul cifrului este cuprinsă între 1 și 99999.
- În cazul în care nu poate fi descoperită o secvență de atingeri, *Iahim Uratrox* nu va mai putea deschide poarta folosind o încărcătură explozivă așa cum ar fi putut face atunci când se afla în fața porții orașului, deoarece ar risca să distrugă prea multe vestigii importante; astfel, cel mai mare secret al *Atlantidei* va fi păstrat pentru totdeauna. Din fericire, după câteva luni, exploratorul s-a întors și a prezentat motivele scufundării *Continentului Pierdut*; se pare că explicația era atât de îngrijorătoare încât s-a decis ca motivele să nu fie cunoscute publicului. Totuși, concluzia evidentă care poate fi trasă este că există întotdeauna o secvență corectă de atingeri care duc la deschiderea porții.
- Poarta se deschide chiar dacă numărul atingerilor nu este minim.
- Dacă există mai multe soluții, doar una trebuie scrisă în fișierul de ieșire.

Exemplu

ATLANTIS.IN	ATLANTIS.OUT
00012	++-+

Timp de execuție: 5 secunde/test

Premianții

Premiul I

Sorin Pahasciuc, Bistrița-Năsăud - cls. a IX-a
Andrei Barabaș, Maramureș - cls. a X-a
Flaviu Pașca, Bistrița-Năsăud - cls. a XI-a
Vencel Bors, Bihor - cls. a XII-a
Paul Rusu, Cluj - cls. a XII-a

Premiul II

Tiberiu Chihorean, Maramureș - cls. a X-a
Nicolae Preda, Cluj - cls. a XI-a
Andrei Markovits, Satu Mare - cls. a XII-a

Premiul III

Bogdan Pănuș, Satu Mare - cls. a XI-a
Adrian Balașko, Sălaj - cls. a XII-a

Târgu-Jiu, județul Gorj

Concursul interjudețean INFO-OLTENIA

În perioada 21-23 februarie 2002 a avut loc la Colegiul Național "Ecaterina Teodoroiu" din Târgu-Jiu cea de-a V-a ediție a concursului interjudețean de informatică "Info-Oltenia". Vă prezentăm în continuare enunțurile problemelor propuse spre rezolvare la acest concurs.

P030207: Imperiu

(clasa a IX-a, concursul individual)

Un mare imperiu din antichitate avea o suprafață de K metri pătrați. În ultima perioadă a existenței sale, imperiul a purtat N războaie pe care le-a pierdut.

După fiecare război pierdut, imperiul a fost nevoit să cedeze câte o suprafață dreptunghiulară de teren.

Se cere să se determine suprafața de teren pe care o deținea imperiul după terminarea războaielor.

Date de intrare

Fișierul text `IMPERIU.IN` conține pe prima linie valoarea K, care indică suprafața imperiului. Pe a doua linie se află numărul N al războaielor pierdute, iar pe următoarele N linii se află perechi de numere naturale care reprezintă lungimea și lățimea suprafeței de teren dreptunghiular. Numerele de pe o linie sunt separate prin spații.

Date de ieșire

În fișierul text `IMPERIU.OUT` se va scrie, pe prima linie, suprafața de teren rămasă după terminarea războaielor.

Restricții

- $1 \le N \le 50$;
- dimensiunile suprafețelor dreptunghiulare sunt numere naturale care conțin cel mult 20 de cifre;
- K este un număr natural care conține cel mult 200 de cifre.

Exemplu

IMPERIU.IN	IMPERIU.OUT
100 2 6 5 4 10	30

Timp de execuție: 1 secundă/test

P030208: Șir de cifre

(clasa a IX-a, concursul individual)

Asupra unui număr natural k se realizează operațiile:

- se calculează suma S a cifrelor sale;
- în cazul în care numărul S are cel puțin două cifre, asupra acestuia se aplică aceeași operație.

În urma acestei prelucrări, pentru orice număr k se obține un număr format dintr-o singură cifră. Acest număr va fi numit în continuare *cifră finală*.

Se consideră două numere naturale m și n. Sarcina voastră este de a scrie un program care determină cifrele finale pentru numerele cuprinse între m și n.

Date de intrare

Fișierul de intrare `SIR.IN` conține o singură linie pe care se află numerele m și n, separate printr-un spațiu.

Date de ieșire

Fișierul de ieșire `SIR.OUT` va conține două linii. Pe prima dintre acestea se vor afla cifrele finale distincte care au fost obținute. Acestea vor fi ordonate crescător și vor fi separate prin câte un spațiu. A doua linie va conține numărul de apariții ale fiecărei cifre finale; al i-lea element de pe această linie va indica numărul de apariții al celei de-a i-a cifre finale de pe prima linie. Numerele de pe această linie vor fi separate prin câte un spațiu.

Restricție

- $1 \le m \le n \le 2^{31}-1$.

Exemplu

SIR.IN	SIR.OUT
15423 15429	1 2 3 6 7 8 9 1 1 1 1 1 1 1

Timp de execuție: 1 secundă/test

Despre concurs...

La această ediție a concursului "Info-Oltenia" au participat 34 de elevi din județele Dolj, Gorj, Mehedinți, Olt și Vâlcea.

Acesta este primul concurs interjudețean de programare care s-a desfășurat în județul Gorj. Concursul s-a ridicat la standardele olimpiadei naționale; dificultatea problemelor a fost destul de ridicată, iar evaluarea a fost automată.

În timpul concursului concurenții au avut la dispoziție o rețea de 25 de calculatoare dotate cu procesoare Intel Pentium 4 având frecvența de 1,5 GHz.

Au fost două secțiuni: o întrecere individuală, și una pe echipe.

Concursul a reprezentat un excelent prilej de pregătire pentru elevii care participă la olimpiada de informatică din acest an.

Gazeta de Informatică a fost reprezentată la acest concurs de Claudiu Soroiu, redactor al revistei noastre.

P030209: Serii

(clasa a X-a, concursul individual)

O firmă de *hardware* își codifică produsele astfel: un cod este format dintr-o secvență de cifre urmată de o secvență de caractere (litere mici ale alfabetului latin). În momentul apariției unui nou produs, firma este interesată de codul următorului produs și de codul produsului anterior. Codurile produselor sunt generate astfel: se generează secvența următoare în ordine lexicografică formată din aceleași caractere ale codului produsului anterior (fără a modifica secvența numerică a codului anterior), iar dacă nu există o secvență următoare se incrementează cu 1 secvența numerică și se generează primul cod în ordine lexicografică, format din aceleași caractere.

Se cere ca pentru un anumit cod dat, să se genereze codul anterior și codul următor al unui produs.

Date de intrare

Fișierul de intrare **SERII.IN** conține o singură linie pe care se alfă codul unui produs.

Date de ieșire

Fișierul de ieșire **SERII.OUT** va conține pe prima linie codul produsului anterior, iar pe a doua linie codul produsului următor.

În cazul în care, pentru un anumit produs, unul dintre cele două coduri nu există, pe linia respectivă se va scrie caracterul `'*'`.

Restricții și precizări

- secvența numerică poate să aibă cel mult 8 cifre;
- secvența de caractere conține cel mult 1000 de caractere.

Exemple

SERII.IN
```
000abc
```
SERII.OUT
```
*
000acb
```

SERII.IN
```
153xabc
```
SERII.OUT
```
153cxba
153xacb
```

SERII.IN
```
2457z
```
SERII.OUT
```
2456z
2458z
```

Timp de execuție: 1 secundă/test

P030210: Vize

(clasa a X-a, concursul individual)

Într-un univers paralel au loc schimbări politice cel puțin bizare. Astfel, țările de pe un anumit continent anulează necesitatea vizelor de intrare în cazul în care este respectată o anumită condiție: două țări numite n_1, respectiv n_2 anulează necesitatea vizelor dacă există cel puțin un cuvânt care conține k caractere și care se poate obține atât prin ștergerea de caractere din n_1, cât și prin ștergerea de caractere din n_2.

De exemplu, pentru $k = 4$, țările `Romania` și `Polonia` anulează vizele deoarece există cuvântul `onia` care se poate obține atât prin ștergerea caracterelor `'R'`, `'m'` și `'a'` din cuvântul `Romania`, cât și prin ștegerea caracterelor `'P'`, `'o'`, `'l'` (sau `'P'`, `'l'`, `'o'`) din cuvântul `Polonia`.

În plus, dacă au fost anulate vizele între țările n_1 și n_2 și au fost anulate vizele și între țările n_2 și n_3, atunci se anulează vizele și între țările n_1 și n_3.

Se cere să se determine grupurile de țări între care se poate circula fără vize.

Date de intrare

Pe prima linie a fișierului de intrare **VIZE.IN** se află numărul n al țărilor și valoarea k. Aceste două numere sunt separate printr-un spațiu. Pe fiecare dintre următoarele n linii se află numele unei țări.

Date de ieșire

Fișierul de ieșire **VIZE.OUT** va conține câte o linie pentru fiecare grup de țări între care se poate circula fără viză. Pe linia corespunzătoare unui grup se vor afla numerele de ordine ale țărilor din acest grup. Numerele de pe o linie vor fi separate prin câte un spațiu. Numerele de ordine ale țărilor pot fi determinate pe baza ordinii în care acestea apar în fișierul de intrare.

Restricții și precizări

- numele unei țări este format dintr-un singur cuvânt care conține cel mult 50 de caractere;
- nu se face distincție între literele mari și literele mici;
- $1 < n < 100$.

Exemplu

VIZE.IN	VIZE.OUT
3 4	1 3
Polonia	2
Cehia	
Romania	

Timp de execuție: 1 secundă/test

P030211: Fibbo

(clasele a XI-a și a XII-a, concursul individual)

Se consideră *șirul lui Fibonacci* definit astfel:

- $F_0 = 0$;
- $F_1 = 1$;
- $F_i = F_{i-1} + F_{i-2}$ pentru $i > 1$.

Să se determine cel mai mic număr strict pozitiv p, astfel încât F_p să fie un element al șirului lui *Fibonacci* care este divizibil cu 2^k.

Date de intrare

Fișierul de intrare **FIBO.IN** conține o singură linie pe care se află numărul k.

Date de ieșire

Fișierul de ieșire **FIBO.OUT** va conține o singură linie pe care se va afla numărul p.

Restricție

- $0 \le k \le 9000$.

Exemple

FIBO.IN	FIBO.OUT
3	6

FIBO.IN	FIBO.OUT
7	96

Timp de execuție: 1 secundă/test

P030212: Prieteni

(clasele a XI-a și a XII-a, concursul individual)

Un grup de n elevi sunt identificați prin numere cuprinse între 1 și n. Între aceștia există relații de prietenie directă. Relația de prietenie este simetrică în sensul că dacă elevul i este prieten cu elevul j, atunci și elevul j este prieten cu elevul i.

Se știe că o informație poate ajunge de la oricare elev la toți ceilalți în condițiile în care un elev o poate comunica doar prietenilor săi.

Un părinte vrea să facă un test; el încearcă să repartizeze elevii în două camere diferite, astfel încât doi prieteni direcți să nu se afle în aceeași cameră.

În cazul în care repartizarea este posibilă trebuie determinate numerele de ordine ale elevilor din prima cameră.

În cazul în care repartizarea nu este posibilă trebuie determinate perechile de elevi care ar trebui să nu fie prieteni pentru ca repartizarea să fie posibilă, dar informația să poată ajunge de la oricare elev la toți ceilalți în condițiile în care ea poate fi transmisă doar prietenilor direcți.

Date de intrare

Fișierul de intrare **PRIETENI.IN** conține pe prima linie valoarea n. Pe fiecare dintre următoarele n linii se află câte două numere, separate printr-un spațiu, care vor identifica doi prieteni.

Date de ieșire

Dacă repartiția este posibilă, prima linie a fișierului de ieșire **PRIETENI.OUT** va conține valoarea 1. Pe a doua linie se vor afla numerele de ordine ale elevilor din prima cameră, iar pe a treia linie se vor afla numerele de ordine ale elevilor din a doua cameră. Numerele de pe o linie vor fi separate prin câte un spațiu.

În cazul în care repartiția nu este posibilă, prima linie a fișierului de ieșire va conține valoarea 0. Cea de-a doua linie va conține numărul k al perechilor de elevi care ar trebui să nu fie prieteni. Pe fiecare dintre următoarele k linii se vor afla câte două numere, separate printr-un singur spațiu, care vor identifica doi elevi care ar trebui să nu fie prieteni.

Restricții

- $2 < n \le 50$.

Exemple

PRIETENI.IN	PRIETENI.OUT
6	1
1 2	1 3 5
2 3	2 4 6
4 1	
3 4	
4 5	
5 6	

PRIETENI.IN	PRIETENI.OUT
6	0
1 2	1
2 3	1 3
4 1	
3 4	
4 5	
5 6	
1 3	

Timp de execuție: 1 secundă/test

P030213: Florărie

(clasele a IX-a și a X-a, concursul pe echipe)

Gigel lucrează la o florărie de două zile și a primit marfă, ca urmare a mai multor comenzi făcute de colega din cealaltă tură. Din păcate, comenzile au fost mai multe, au venit toate în același timp și trebuie livrate imediat, iar *Gigel*

trebuie să se descurce în această situație, neștiind exact pentru cine sunt florile, dar are la dispoziție următoarele informații: au sosit trei pachete dintre care unul poate fi împărțit în buchete de câte 7 sau 11 flori, unul în buchete de câte 13 flori, dar nu și de 5 flori, iar altul în buchete de câte 5 flori, dar nu și de 13 flori. Dacă *Gigel* reușește să identifice cele trei pachete, el va găsi și destinatarii. Așadar, el este nevoit să apeleze la un calculator de buzunar, fiind singurul pe care-l are. Din nefericire, el constată că nici acesta nu-i poate fi de ajutor, deoarece nu permite folosirea unor numere suficient de mari pentru a putea face împărțirile (poate realiza operații cu numere de cel mult 20 cifre). Dacă numerele ar avea cu trei cifre mai puțin, ar fi putut să efectueze oprațiile folosind calculatorul. *Gigel* trebuie să rezolve problema repede și nu are alt calculator la dispoziție. Ajutați-l să rezolve problema!

Date de intrare

Fișierul de intrare **FLORI.IN** conține o singură linie pe care se vor afla trei numere, separate prin câte un spațiu, care vor indica numărul florilor din cele trei pachete.

Date de ieșire

Rezultatele vor fi afișate în fișierul **FLORI.OUT** astfel:

- pe prima linie se va afla numărul florilor care se împart în buchete de 7 sau 11 flori;
- pe a doua linie se va afla numărul de flori care pot fi împărțite în buchete de 5 flori;
- pe a treia linie se va afla numărul de flori care pot fi împărțite în buchete de 13 flori.

Exemplu

FLORI.IN

```
12057493063 6424482779 1435546875
```

FLORI.OUT

```
12057493063
1435546875
6424482779
```

Timp de execuție: 1 secundă/test

P030214: Puncte în plan

(clasele a IX-a și a X-a, concursul pe echipe)

Se consideră n puncte în plan. Să se determine patrulaterul convex ale cărui vârfuri se regăsesc printre punctele date și în interiorul căruia se află cât mai multe puncte.

Date de intrare

Fișierul de intrare **PUNCTE.IN** conține mai multe linii pe care se vor afla câte două numere întregi, separate prin câte un spațiu, care reprezintă coordonatele punctelor.

Date de ieșire

În cazul în care există un astfel de patrulater, fișierul de ieșire **PUNCTE.OUT** va conține cinci linii. Pe primele patru se vor afla coordonatele punctelor care reprezintă vârfurile patrulaterului, în sensul acelor de ceasornic. Pe cea de-a cincea linie se va scrie numărul punctelor din interiorul patrulaterului. Dacă problema nu are soluție, fișierul de ieșire va conține o singură linie pe care se va afla caracterul `'*'`.

Restricții și precizări

- $4 \le n \le 100$;
- coordonatele punctelor sunt numere întregi cuprinse în intervalul [-200, 200].
- un punct este considerat a fi situat în interiorul patrulaterului chiar dacă se află pe o latură a acestuia.

Exemplu

PUNCTE.IN

```
1 6
1 4
3 3
6 4
2 1
11 5
7 3
3 2
7 6
7 1
4 5
```

PUNCTE.OUT

```
2 1
1 6
7 6
7 1
5
```

Timp de execuție: 1 secundă/test

P030215: Produs

(clasele a XI-a și a XII-a, concursul pe echipe)

Fie un număr real n; să se determine o valoare k și k numere reale $a_1, a_2, ..., a_k$ astfel încât $a_1 + a_2 + ... + a_k = n$ și $a_1 \cdot a_2 \cdot ... \cdot a_k$ este maxim.

Date de intrare

Fișierul de intrare **PRODUS.IN** conține o singură linie pe care se va afla valoarea n.

Date de ieșire

Fișierul de ieșire **PRODUS.OUT** va conține pe prima linie numărul natural k. Pe cea de-a doua linie se vor afla numerele $a_1, a_2, ..., a_k$, separate prin câte un singur spațiu. Acestea vor fi afișate cu cinci zecimale exacte.

Restricții

- $2 \le n \le 5 \cdot 10^{10}$

Exemple

PRODUS.IN

```
4
```

PRODUS.OUT

```
1
4.00000
```

PRODUS.IN

```
5
```

PRODUS.OUT

```
2
2.50000 2.50000
```

PRODUS.IN	PRODUS.OUT
6	2 3.00000 3.00000

Timp de execuție: 1 secundă/test

P030216: Parcele

(clasele a XI-a și a XII-a, concursul pe echipe)

Un țăran dintr-un sat pierdut în *Oltenia*, fiind destul de înaintat în vârstă și neavând urmași, s-a gândit să-și împartă lotul de pământ consătenilor. Acest lot este alcătuit din tarlale cultivate cu grâu și porumb, are forma unui pătrat și seamănă cu o tablă de șah de $n \times n$ tarlale ("albele" fiind tarlalele de grâu, iar "negrele" cele de porumb).

Țăranul își propune să facă o împărțire astfel încât să fie respectate următoarele condiții:

- fiecare consătean să primească o suprafață de pământ de formă dreptunghiulară;
- fiecare asemenea suprafață de pământ trebuie să conțină un număr întreg de tarlale;
- în fiecare suprafață de pământ numărul de tarlale de grâu trebuie să fie egal cu numărul tarlalelor de porumb;
- numărul consătenilor care primesc pământ este maxim;
- nu trebuie să existe doi consăteni care să primească aceeași suprafață de pământ.

Datorită faptului că țăranul nu dispune de cunoștințele matematice necesare realizării împărțirii, va trebui să îl ajutați.

Date de intrare

Fișierul de intrare **LOT.IN** conține o singură linie pe care se află valoarea lui n.

Date de ieșire

Fișierul de ieșire **LOT.OUT** va conține, pe prima linie, numărul k al consătenilor care vor primi pământ în urma divizării lotului. Cea de-a doua linie va conține k valori care vor indica suprafețele (numărul de tarlale) pe care le vor primi cei k consăteni.

Pe următoarele n linii se vor afla câte n numere care vor descrie o modalitate de a dispune respectivele suprafețe în lotul inițial.

O linie a lotului de pământ va ocupa o linie a fișierului, fiecare tarla a lotului de pământ inițial va fi reprezentată printr-un număr egal cu aria suprafeței căreia îi aparține; două astfel de numere vor fi despărțite printr-un spațiu.

Dacă nu există nici o descompunere validă, în fișierul de ieșire se va scrie mesajul "IMPOSIBIL".

Restricție

- $2 \le n \le 150$

Exemplu

LOT.IN	LOT.OUT
6	5 2 4 6 8 16 8 8 8 8 2 6 8 8 8 8 2 6 4 16 16 16 16 6 4 16 16 16 16 6 4 16 16 16 16 6 4 16 16 16 16 6

Timp de execuție: 1 secundă/test

Premianții

Clasa a IX-a, individual

Andrei Gheorghe, Olt - premiul I
Radu Vita, Mehedinți - premiul II
Dan Mitroi, Mehedinți - premiul III

Clasa a X-a, individual

Nicolae Tănase, Olt - premiul I
Andrei Homescu, Gorj - premiul II
Alexandru Coșoi, Olt - premiul III

Clasa a XI-a, individual

Silviu Udrea, Gorj - premiul I
Iulian Rădulescu, Vâlcea - premiul II
Bogdan Drăgușin, Olt - premiul III

Clasa a XII-a, individual

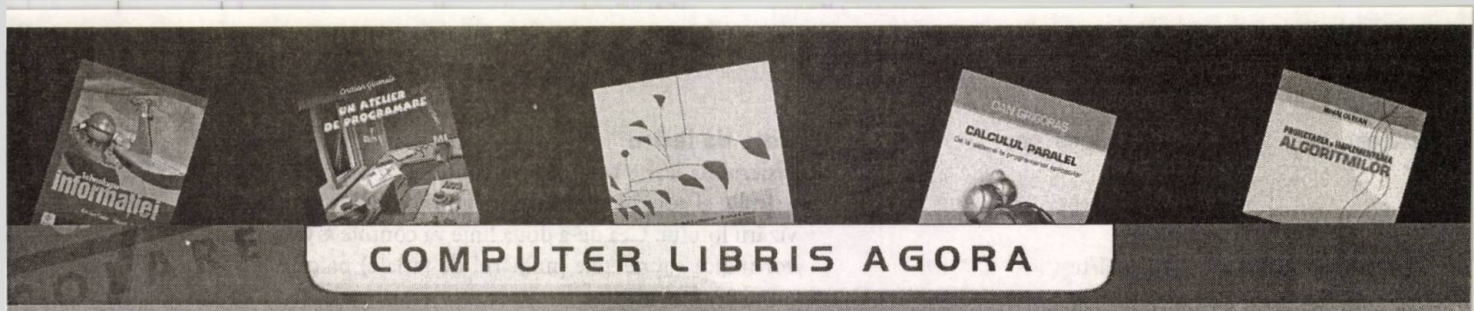
Patrick Bălulescu, Mehedinți - premiul I
Cătălin Dorobanțu, Olt - premiul II
Cristian Răuț, Dolj - premiul III

Clasele a IX-a și a X-a, echipe

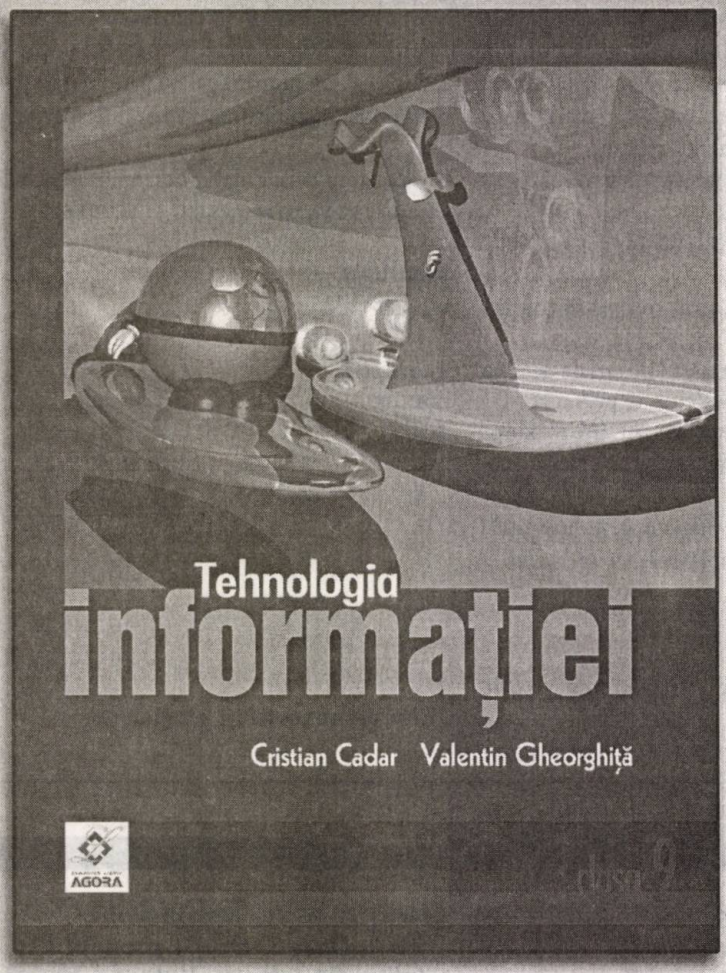
Andrei Homescu și Emanuel Udrea, Gorj - premiul I
Dan Mitrui și Radu Vita, Mehedinți - premiul II
Andrei Gheorghe și Alexandru Coșoi, Olt -premiul III

Clasele a XI-a și a XII-a, echipe

Bogdan Drăgușin și Cătălin Dorobanțu, Olt - premiul I
Patrick Bălulescu și Cristian Popescu, Mehedinți - premiul II
Radu Băluță și Silviu Udrea, Gorj - premiul III



COMPUTER LIBRIS AGORA



Lucrarea de față își propune să pregătească cititorul în scopul utilizării unui calculator personal. Ea urmărește programa școlară, aprobată de Ministerul Educației Naționale, pentru disciplina "Tehnologia informației", disciplină în planul de învățământ a claselor a IX-a la profilul teoretic și pedagogic. Manualul corespunde și programei de la disciplina "Informatică - Tehnologii asistate pe calculator", profilul tehnologic.

Pentru parcurgerea acestei cărți nu sunt necesare cunoștințe prealabile în domeniu, termenii și noțiunile prezentate sunt explicate detaliat în carte. Exemplele sunt sugestive, cartea nu este încărcată inutil cu informații care depășesc programa amintită.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

Suceava, județul Suceava

Linfo@SV

Pe data de 9 februarie 2002 s-a desfășurat la Liceul de Informatică "Spiru Haret" din Suceava prima ediție a concursului de programare și creativitate în domeniul software "Linfo@SV". Vă prezentăm în continuare enunțurile problemelor propuse spre rezolvare la secțiunea de programare a acestui concurs.

P030217: Biletul norocos

(clasa a IX-a)

Un hoț găsește un bilet în care sunt înscrise mișcările pe care trebuie să le facă pentru a deschide seiful singurei bănci din orașul său. Biletul conține un text sub forma unui șir care conține cifre urmate de o literă *s* sau *d*, după cum trebuie rotit butonul la stânga sau la dreapta.

De exemplu, șirul `4s5d1s` indică faptul că butonul trebuie rotit cu patru poziții spre stânga, apoi cu cinci poziții spre dreapta și, în final, cu o poziție spre stânga.

În jurul butonului seifului sunt imprimate cifre cuprinse între 0 și 9, iar ordinea acestora (în sens orar) este cunoscută. De asemenea, este cunoscută poziția inițială a butonului.

Pe baza biletului găsit de hoț și a marcajelor din jurul butonului, va trebui să determinați codul necesar pentru a deschide seiful.

Date de intrare

Fișierul de intrare **`BILET.IN`** conține pe prima linie șirul de mișcări care trebuie efectuate asupra butonului. Pe a doua linie se află mai multe numere cuprinse între 0 și 9, separate printr-un spațiu, iar linia a treia conține un număr care reprezintă poziția curentă a încuietorii seifului.

Date de ieșire

Fișierului de ieșire **`BILET.OUT`** va conține o singură linie pe care se va afla codul seifului.

Exemplu

`BILET.IN`

```
4s1d5s1s2d
9 1 7 8 0 4 2 3 6 5
2
```

`BILET.OUT`

```
36870
```

Timp de execuție: 1 secundă/test

P030218: Zile de naștere

(clasa a IX-a)

EA și *EL* sunt prieteni de mai mult timp, dar *EL* nu știe care este ziua ei de naștere. Ca să nu fie prins pe picior greșit, *EL* încearcă să afle de la prietenele ei răspunsul, dar nimeni nu vrea să-l ajute. Singura posibilitate este să o întrebe direct pe *EA*; fiind elevă la *Liceul de Informatică*, *EA* îi propune următorul joc:

Îți spun un număr de cinci cifre care reprezintă data mea de naștere codificată pe 16 biți astfel:

- *primii șapte biți (cei mai semnificativi) reprezintă ultimele două cifre ale anului;*
- *următorii patru biți reprezintă luna;*
- *ultimii cinci biți reprezintă ziua.*

Tu va trebui să îmi spui care este ziua mea de naștere sub forma zi, lună, an sau, dacă am încercat să te păcălesc, că numărul nu poate fi o dată de naștere.

EL este de acord, dar vrea, ca la sfârșit, *EA* să determine numărul care corespunde zilei lui de naștere, zi pe care el nu o ține secretă.

Date de intrare

Fișierul de intrare **`ZILE.TXT`** conține pe prima linie numărul care codifică data ei de naștere, iar pe a doua linie data lui de naștere sub forma: zi, lună, an.

Date de ieșire

Pe prima linie a fișierului de ieșire **`REZ.TXT`** se va afla data ei de naștere sub forma zi, lună, an, dacă numărul dat corespunde unei date corecte, sau cuvântul `GRESIT`, în caz contrar. A doua linie va conține un singur număr care codifică data lui de naștere.

Exemplu

`ZILE.TXT`

```
43911
4 1 82
```

`REZ.TXT`

```
12 7 85
42020
```

Timp de execuție: 1 secundă/test

Despre concurs...

Concursul Linfo@SV s-a desfășurat pe plan local, fiind invitați elevi de la liceele județului Suceava. Sponsorii concursului au fost Fundația Cooperation Humanitaire Luxembourg ONG și grupul de firme WARPNET.

Concursul a avut două secțiuni: una de programare și una de creativitate în domeniul software.

La secțiunea dedicată programării au participat 58 de elevi.

P030219: Anagrame

(clasa a X-a)

Se consideră o propoziție terminată prin punct. Aceasta este alcătuită din cuvinte formate numai din litere mari. Cuvintele sunt separate între ele prin unul sau mai mulți separatori care pot fi spații sau virgule.

Știind că propoziția are cel mult 30 de cuvinte, se cere să se determine grupurile de cuvinte în care fiecare membru reprezintă o anagramă a oricărui alt membru al aceluiași grup. De asemenea, trebuie determinat numărul de cuvinte din fiecare grup. Numărul grupurilor determinate trebuie să fie minim. Cuvintele care fac parte dintr-un anumit grup trebuie să fie distincte, iar "grupurile" formate dintr-un singur cuvânt nu vor fi luate în considerare.

Date de intrare

Fișierul de intrare **IN.TXT** va conține o singură linie pe care se află propoziția.

Date de ieșire

Fișierul de ieșire **OUT.TXT** va conține mai multe linii, pe fiecare dintre ele aflându-se cuvinte care aparțin aceluiași grup, separate printr-un spațiu și, la sfârșitul fiecărui rând, un număr întreg strict pozitiv care reprezintă numărul de cuvinte din grup.

Restricții și precizări

- un cuvânt are cel mult 16 litere;
- anagramă a unui cuvânt este orice alt cuvânt obținut prin permutarea literelor primului.

Exemplu

IN.TXT

```
CAR LABIRINT STROP ACT ARC BRILIANT PROST
                                    RAC TAC CAT.
```

OUT.TXT

```
CAR ARC RAC 3
LABIRINT BRILIANT 2
STROP PROST 2
ACT TAC CAT 3
```

Timp de execuție: 1 secundă/test

P030220: Internet Café

(clasa a X-a)

WarpNet Internet Café dispune de un număr de N^2 calculatoare așezate într-o rețea de formă pătratică. Pentru a răspunde cerințelor de confidențialitate solicitate de unii clienți, între unele posturi de lucru s-au montat P panouri separatoare. Panourile pot fi instalate între două calculatoare așezate alăturat sau unul în spatele celuilalt.

Determinați care este numărul de posibilități de aranjare a celor M clienți, astfel încât fiecare dintre ei să fie sigur că nu este "spionat" de nici un alt client. "Spionajul" poate fi, eventual, practicat numai între posturi de lucru aflate pe aceeași linie sau coloană.

Date de intrare

Fișierul de intrare **INTERNET.IN** conține pe prima linie trei numere întregi pozitive care reprezintă valorile N, M și P, separate printr-un spațiu. Pe următoarele P linii se află coordonatele calculatoarelor între care există panouri despărțitoare, separate printr-un spațiu.

Date de ieșire

Fișierul de ieșire **INTERNET.OUT** va conține o singură linie pe care se va afla numărul de posibilități de așezare al celor M clienți, astfel încât nici unul să nu fie "spionat".

Exemplu

INTERNET.IN	INTERNET.OUT
3 4 4	168
1 2 1 3	
1 2 2 2	
2 1 2 2	
2 2 3 2	

Timp de execuție: 5 secunde/test

P030221: Cifre

(clasele a XI-a și a XII-a)

Se cunosc trei cifre, nu neapărat distincte, cu valori cuprinse între 1 și 5. Cu aceste trei cifre, folosite o singură dată, se pot construi expresii matematice astfel:

- operanzii din expresia creată sunt formați din una sau două cifre dintre cele trei cifre;
- între oricare operanzi se pot folosi operații de adunare, scădere, înmulțire, împărțire și ridicare la putere.

Se cere să se determine valoarea cea mai mare pe care o poate avea una din expresiile construite conform regulilor precedente.

Date de intrare

Fișierul de intrare **CIFRE.TXT** conține pe prima linie cele trei cifre, separate printr-un singur spațiu.

Date de ieșire

Fișierul de ieșire **MAX.TXT** va conține cel mai mare număr obținut ca valoare a uneia din expresiile construite.

Exemplu

CIFRE.TXT	MAX.TXT
1 3 2	10460353203 (*rezultatul expresiei* 3^{21})

Timp de execuție: 1 secundă/test

P030222: Insula furnicilor

(clasele a XI-a și a XII-a)

Pe *Insula Furnicilor* era vremea strânsului recoltei. Toate furnicile erau ocupate să-și adune cât mai multe provizii pentru iarnă. Deodată vântul a început să bată și toată recolta furnicilor risca să fie luată de vânt. Regina *ALT-A* a convocat repede consiliul de coroană pentru situația de urgență creată și le-a prezentat harta insulei, pe care erau reprezentate toate drumurile care pleacă de la câmpul cu recolta, la depozitul aflat în mușuroi. Harta era desenată pe hârtie milimetrică având n intersecții. Pentru fiecare drum de pe hartă era cunoscut numărul maxim de boabe care puteau fi transportate simultan.

Regina le-a cerut miniștrilor săi să găsească o strategie de transport a recoltei de pe câmp, fără a modifica rețeaua de drumuri (din lipsă de timp, deoarece se apropia furtuna), astfel încât într-un singur transport cantitatea de provizii a mușuroiului să fie maximă. Pentru reușita operației, regina *ALT-A* a ordonat ca transportul recoltei să fie supravegheat de însuși ministrul agriculturii, *Marele DOT*, în fiecare intersecție de pe traseu să fie prezent câte un *Furnicar Public*, pentru evitarea blocajelor de pe traseu, iar la hambarul mușuroiului regina va veghea activitatea de depozitare.

Furnicarii Publici vor avea drept sarcină de serviciu calcularea cantității totale de boabe sosite și plecate din intersecție. Fiecare *Furnicar Public* va primi drept recompensă, la sfârșit, 5% din cantitatea totală de boabe care a trecut prin intersecția pe care acesta a supravegheat-o.

Date de intrare

Fișierul de intrare **INSULA.IN** are următoarea structură:

- prima linie conține un număr N care reprezintă numărul de intersecții (inclusiv câmpul cu recolta și depozitul din mușuroi);
- a doua linie conține numerele S și D, separate printr-un spațiu, care reprezintă pozitia câmpului cu recolta, respectiv poziția depozitului central al mușuroiului;
- următoarele linii conțin triplete de numere i, j, c, separate printr-un spațiu, cu semnificația că există un drum de la intersecția i la intersecția j, iar pe acest drum pot fi transportate maxim c boabe.

Date de ieșire

Fișierul de ieșire **RECOLTA.OUT** va avea următoarea structură:

- prima linie va conține un număr C_{max} care reprezintă cantitatea maximă de hrană transportată;
- a doua linie va conține n - 2 numere, separate între ele printr-un singur spațiu, care reprezintă recompensele celor n - 2 funcționari de pe traseu;
- următoarele linii vor conține fiecare câte trei numere i, j și r, separate printr-un singur spațiu, care au următoarea semnificație: pe drumul de la i la j s-au transportat r boabe.

Restricții și precizări

- $3 \le n \le 500$;
- pentru a nu fi nevoită să taie boabele, regina a hotărât ca recompensa *Furnicarilor Publici* să fie formată din cel mai mare număr întreg de boabe mai mic decât valoarea pe care ar fi trebuit să o primească fiecare.

Exemplu

INSULA.IN	RECOLTA.OUT
5	170
1 4	5 4 6
1 2 100	1 2 100
2 5 100	1 5 70
1 5 130	2 3 60
2 3 60	2 5 40
5 3 90	3 4 80
3 4 80	5 3 20
5 4 90	5 4 90

Timp de execuție: 2 secunde/test

Premianții

Premiul I

Daniel Ghiță, clasa a IX-a
Alexandru Artimon, clasa a X-a
Ștefăniță Fechete, clasele a XI-a și a XII-a

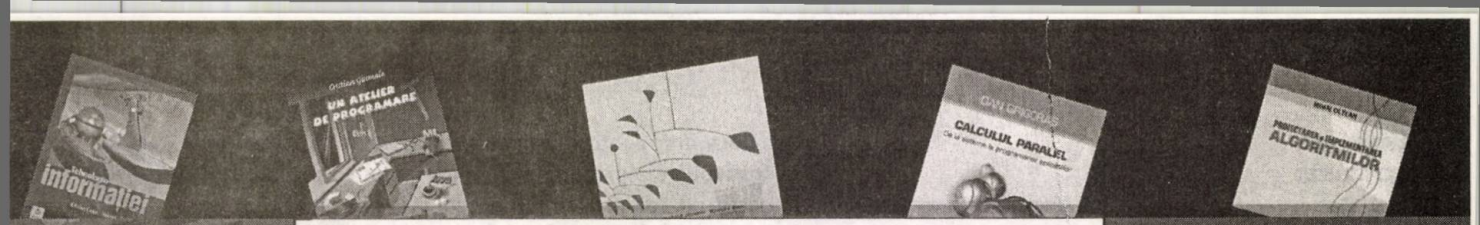
Premiul II

Paul Grigoraș, clasa a IX-a
Cosmin Pupăză, clasa a X-a
Silviu Niculiță, clasele a XI-a și a XII-a

Premiul III

Lucian Nistor, clasa a IX-a
Alexandru Burciu, clasa a X-a
Andrei Mihai, clasele a XI-a și a XII-a

Concurs de programare și creativitate în domeniul software
Info@Su 2002



COMPUTER LIBRIS AGORA

Cartea este o culegere de studii de caz, construite pe temelia problemelor concursurilor Southeastern European Regional Programming Contest din perioada 1995-1999, concursuri organizate sub egida Association for Computing Machinery.
Cartea se adresează atât începătorilor, cât și celor cu experiență în domeniul programării, în încercarea de a contura prin exemple imaginea unei lumi diverse și pasionante, dar pretențioase, aceea a înzestrării mașinii de calcul cu capacitatea de a rezolva corect și eficient.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

Bursele AGORA

A treia etapă a concursului de programare Bursele Agora s-a încheiat. În cadrul acestui număr vă vom prezenta soluțiile problemelor de la această rundă, clasamentul etapei, precum și clasamentul general după primele trei etape.

A treia rundă

Pentru prima dată la această ediție a concursului nostru, unul dintre concurenți a reușit să obțină toate cele 300 de puncte puse în joc la o rundă. Performanța a fost obținută de *Silvestru-Cosmin Negrușeri*, din *Bistrița*.

Concitadinul său, *Adrian Cârcu*, ar fi putut obține și el punctajul maxim dar, datorită unei erori banale, a trebuit să se mulțumească doar cu 200 de puncte. Soluția sa de la problema *Cub* ar fi obținut 100 de puncte dacă datele de intrare ar fi fost citite din fișierul `CUBE.IN` (în loc de `CUBES.IN`) și rezultatele ar fi fost scrise în fișierul `CUBE.OUT` (în loc de `CUBES.OUT`).

Profităm de "ocazie" pentru a vă atenționa din nou că trebuie să fiți foarte atenți la specificațiile problemelor și la modul în care trebuie construită arhiva pe care o trimiteți.

Cele mai mari punctaje obținute la această rundă sunt:

1. Silvestru-Cosmin Negrușeri, Bistrița	*300*
2. Mugurel-Ionuț Andreica, București	*296*
3. Octavian-Daniel Dumitran, București	*264*
4. Adrian Balașko, Zalău	*252*
5. Radu Dondera, București	*248*
6. Bogdan Nicolae, Sibiu	*236*
7. Liviu-Cosmin Andreicuț, Bacău	*234*
8. Radu-Andrei Ștefan, Brașov	*231*
9. Csaba András, Oradea	*223*
10. Ștefăniță Fechete, Suceava	*214*
11. Andrei Giurgiu, București	*203*
12. Tiberiu Dăneț, București	*202*
13. Adrian-Nicolae Cârcu, Bistrița	*200*
13. Maximilian Machedon, București	*200*
13. Flaviu Pașca, Bistrița	*200*
16. Paul Rusu, Cluj-Napoca	*197*
17. Andrei Vancea, Cluj-Napoca	*184*
18. Victor-Marius Costan, București	*175*
19. Claudiu Raicu, Pitești	*170*
20. Csaba Pătcaș, Oradea	*163*
20. Andrei-Vasile Csibi, Cluj-Napoca	*163*

Cu excepția problemei *Scânduri*, la care au obținut punctajul maxim doar doi concurenți, la celelalte două probleme au existat o mulțime de participanți care au obținut 100 de puncte. Acest fapt demonstrează că, în general, problemele de la această rundă nu au fost foarte dificile, un punctaj de 200 de puncte fiind relativ ușor de obținut.

Datele de test folosite pentru testare, precum și variante ale ieșirilor corecte, pot fi descărcate folosind *link*-urile de pe paginile dedicate celor trei probleme. Adresele acestor pagini sunt:

- www.ginfo.ro/concurs/templu.shtml;
- www.ginfo.ro/concurs/scanduri.shtml;
- www.ginfo.ro/concurs/cub.shtml.

La mulți ani!

Andrei Dulceanu, Pașcani - 1 III 1985
Arcadie Crăcan, Iași - 2 III 1984
Radu Ceucă, Simeria - 3 III 1986
Ovidiu Pintican, Tg. Mureș - 5 III 1981
Alina Fețnic, Apahida - 5 III 1985
Andrei David, Râmnicu Vâlcea - 6 III 1982
Emil-Adrian Șpiac, Arad - 6 III 1985
Cezar-Florin Belghiru, Alexandria - 7 III 1984
Nicolae-Gheorghe Cojocaru, Nanov - 8 III 1984
József Csog, Sfântu Gheorghe - 8 III 1985
Emilian Miron, Bacău - 9 III 1985
Zoltán Szilágyi, Sfântu Gheorghe - 9 III 1985
Csaba Pătcaș, Oradea - 14 III 1983
Octavian-Daniel Dumitran, București - 15 III 1983
Ion Savin, Sighișoara - 17 III 1984
Bogdan-Vasile Hârjoc, Cugir - 18 III 1983
Andrei Paponiu, Drobeta-Turnu Severin - 18 III 1985
Silvia Copil-Crișan, Arad - 19 III 1985
Mihai Voinescu, Târgu-Jiu - 22 III 1982
Liviu-Laurențiu Iacob, București - 22 III 1984
Alexandru Munteanu, Roman - 22 III 1989
Andrei-Eugeniu Ioniță, Slatina - 25 III 1986

Runda 3, problema 1

TEMPLU

Această problemă a fost rezolvată corect de 19 dintre participanții la a treia rundă a concursului *Bursele Agora*, așadar soluția ei nu era foarte greu de găsit.

Problema este o adaptare a uneia dintre problemele de rezervă de la ediția 2001 a *Olimpiadei Internaționale de Informatică*, al cărui autor este ***Janne Kujala***.

Enunțul original era următorul: *se consideră un șir de numere binare; folosind acest șir se construiește o matrice în care prima linie este dată de șirul considerat, iar fiecare dintre următoarele linii sunt obținute prin permutarea la stânga cu o poziție a elementelor din linia precedentă; în continuare, liniile matricei obținute sunt sortate lexicografic (valoarea 0 precede valoarea 1); dându-se elementele de pe ultima coloană a matricei obținute după sortare, să se determine elementele de pe prima linie a acestei matrice.*

Soluția pe care o vom prezenta în continuare este preluată din cartea *IOI '01 Competition*, care a apărut după terminarea olimpiadei.

Algoritmul este foarte simplu și constă în trei pași:

- se numără elementele cu valoarea 0 și elementele cu valoarea 1 de pe ultima coloană și, pe baza rezultatului numărării, se construiește prima coloană a matricei; în continuare vom presupune că avem p elemente cu valoarea 0 și q elemente cu valoarea 1.
- se creează un șir a astfel:
  - pentru $i \le p$, elementul a_i va avea ca valoare linia pe care se află cel de-al i-lea element cu valoarea 0 de pe ultima coloană;
  - pentru $j \le q$, elementul a_{p+j} va avea ca valoare linia pe care se află cel de-al j-lea element cu valoarea 1 de pe ultima coloană;
- începând cu prima linie, se parcurg liniile folosind vectorul construit a (de la linia k se trece la linia a_k) și sunt reconstituite elementele de pe prima linie pe baza valorilor de pe ultima coloană.

În cele ce urmează vom demonstra corectitudinea algoritmului prezentat. Numin ***succesor*** al unui șir, șirul obținut printr-o permutare cu o poziție la stânga a elementelor șirului.

Pentru a demonstra corectitudinea algoritmului, va trebui să arătăm că șirul a reprezintă exact această relație de succesiune între șiruri. După demonstrarea acestui fapt, este evident că al treilea pas al algoritmului reconstituie corect elementele de pe prima linie.

Considerăm liniile (din matricea obținută după sortare) care încep cu un element care are valoarea 0. Deoarece, aceste șiruri sunt ordonate în matricea sortată, după efectuarea unei permutări la stânga cu o poziție, șirurile obținute (care se termină cu un element care are valoarea 0) sunt și ele sortate.

Aceeași proprietate este valabilă și pentru liniile care încep cu un element care are valoarea 1.

Deoarece aceste proprietăți sunt echivalente cu relația de succesiune, corectitudinea algoritmului este demonstrată.

Vom demonstra în continuare că timpul de execuție al algoritmului este unul liniar. Pentru aceasta vom arăta că fiecare dintre cei trei pași se efectuează în timp liniar.

Numărarea elementelor de pe ultima coloană se realizează printr-o singură parcurgere, deci liniaritatea este evidentă.

Inițializarea tabloului a necesită doar o singură parcurgere a acestuia și a ultimei coloane, deci timpul este liniar și în acest caz.

Cel de-al treilea pas se execută și el în timp liniar deoarece la fiecare tranziție se determină câte un element de pe prima linie a matricei sortate.

Codul sursă al programului este disponibil pentru *download* la adresa **www.ginfo.ro/concurs/templu.shtml**.

Au rezolvat corect:

Adrian Airinei, Bacău
Csaba András, Oradea
Mugurel-Ionuț Andreica, București
Liviu-Cosmin Andreicuț, Bacău
Adrian Balașko, Zalău
Adrian-Nicolae Cârcu, Bistrița
Victor-Marius Costan, București
Andrei-Vasile Csibi, Cluj-Napoca
Radu Dondera, București
Octavian-Daniel Dumitran, București
Ștefăniță Fechete, Suceava
Andrei Giurgiu, București
Maximilian Machedon, București
Silvestru-Cosmin Negruțeri, Bistrița
Bogdan Nicolae, Sibiu
Flaviu Pașca, Bistrița
Paul Rusu, Cluj-Napoca
Radu-Andrei Ștefan, Brașov
Andrei Vancea, Cluj-Napoca

Runda 3, problema 2

SCÂNDURI

Aceasta a fost cea mai dificilă problemă propusă spre rezolvare la cea de-a treia rundă a ediției 2001/2002 a concursului *Bursele Agora*. Doar doi dintre concurenții care au trimis soluții au reușit să obțină 100 de puncte; este vorba de ***Cosmin-Silvestru Negruşeri*** și ***Adrian Cârcu***, ambii din *Bistrița*. Foarte aproape de a obține punctajul maxim a fost ***Mugurel-Ionuț Andreica*** din *București*; el a obținut 96 de puncte.

Problema ***Scânduri*** face parte din categoria problemelor *NP*-complete, deci găsirea unui algoritm polinomial pentru rezolvarea ei ar însemna demonstrarea celebrei probleme cu care se confruntă cei care studiază algoritmica din punct de vedere teoretic: $P \equiv NP$.

Conform așteptărilor, nu a fost găsită nici o soluție polinomială, deci nici unul dintre concurenți nu poate, deocamdată, intra în posesia premiului de un milion de dolari care va fi oferit celui care va găsi o soluție polinomială pentru o problemă *NP*-completă sau va demonstra că o astfel de soluție nu poate exista.

În continuare vom prezenta modul în care va trebui construit algoritmul exponențial de rezolvare a acestei probleme care să se încadreze în limita de timp impusă.

Pentru a determina soluția vom folosi metoda *backtracking*. Restricțiile impuse în enunț (cel mult cinci scânduri cu aceeași lungime și cel mult cinci bucăți cu aceeași lungime) permit scrierea de programe care să găsească soluția în mai puțin de cinci secunde.

Pentru a micșora cât mai mult posibil spațiul de căutare a soluțiilor va trebui să determinăm câte posibilități de reconstituire a unei scânduri există. Operația poate fi realizată foarte simplu în timp pătratic. Ulterior, vom ordona scândurile în funcție de numărul posibilităților de reconstituire.

Apoi, la fiecare pas al algoritmului care folosește metoda *backtracking* vom considera scândurile rămase în ordinea din șirul sortat. După alegerea a două bucăți care, prin lipire, duc la obținerea unei scânduri, numărul posibilităților de reconstituire se schimbă. Noile valori pot fi obținute în timp liniar. Va urma o reordonare a șirului care trebuie realizată în timp liniar-logaritmic.

În momentul în care am reconstituit toate scândurile, vom afișa soluția.

O implementare a algoritmului prezentat, care optimizează cât mai mult posibil viteza cu care sunt realizate operațiile de la fiecare pas, ar fi obținut toate cele 100 de puncte.

Dacă nu se reușea găsirea unui algoritm performant, puteau fi folosite diverse metode pentru obținerea a cât mai multe puncte.

Datorită faptului că se acordau punctaje parțiale în cazul în care erau reconstituite cel puțin trei sferturi dintre scânduri, este evident faptul că, de fiecare dată când se găsește o soluție mai bună (cu mai multe scânduri reconstituite) decât cea mai bună găsită anterior, atunci aceasta este păstrată. La expirarea intervalului de timp admis pentru execuția programului va fi afișată cea mai bună soluție găsită până în acel moment.

Chiar și unele versiuni mai puțin performante ale algoritmului care folosește metoda *backtracking* ar fi găsit soluțiile pentru testele în care numărul scândurilor era mic. Pentru un număr mai mare de scânduri, în general, puteau fi reconstituite trei sferturi dintre acestea fără ca aceasta să creeze dificultăți deosebite.

O altă posibilitate de a reconstitui cât mai multe puncte era folosirea unor algoritmi probabiliști.

O variantă era de a se încerca reconstituirea unei scânduri, folosind bucățile date (cele două bucăți sunt alese aleator, dar prin lipirea lor trebuie neapărat să fie obținută o scândură), de a elimina bucățile folosite după care se reconstituie altă scândură folosind bucățile rămase etc. În momentul în care nu mai pot fi reconstituite scânduri algoritmul se reia. Procesul se repetă până la expirarea timpului de execuție admis. În final, este aleasă cea mai bună soluție obținută.

Algoritmul are șanse de a reconstitui toate scândurile doar dacă numărul acestora este mic. Pentru valori medii ale numărului de scânduri, există șanse destul de mari să fie reconstituite trei sferturi dintre acestea. În cazul în care numărul scândurilor este mare, șansele de a reconstitui trei sferturi dintre acestea sunt destul de mici.

O altă abordare este folosirea unor metode euristice. Acestea pot combina principiile folosite de algoritmul cu metoda *backtracking* și algoritmul probabilist. Totuși, punctajele obținute de concurenții care au folosit astfel de metode au fost chiar mai mici decât cele obținute de cei care au ales strategiile aleatoare.

Exista și posibilitatea implementării unui algoritm genetic. Chiar dacă această abordare era dificil de implementat, ea putea duce la obținerea punctajului maxim.

Codul sursă al programului care folosește metoda *backtracking* este disponibil pentru *download* la adresa **www.ginfo.ro/concurs/scanduri.shtml**.

Runda 3, problema 3

CUB

Aceasta a fost cea mai simplă problemă a rundei a treia, 20 dintre participanți obținând punctajul maxim. Alți 10 concurenți au obținut peste 85 de puncte.

Problema *Cub* putea fi rezolvată foarte ușor folosind metoda *programării dinamice*.

Considerăm, pe rând, literele cuvântului care trebuie căutat (fie acesta s). La fiecare pas x vom putea determina numărul de apariții al cuvântului format din primele x caractere ale cuvântului dat.

Pentru aceasta, la fiecare pas vom calcula valorile a_{ijk}, unde a_{ijk} indică numărul de apariții ale cuvântului format din primele x litere ale șirului dat, astfel încât elementul din cubulețul de coordonate (i, j, k) a cubului să fie s_x.

Pentru toate căsuțele care nu conțin litera s_x valoarea a_{ijk} va fi 0.

Evident, la primul pas vom avea $a_{ijk} = 1$ pentru toate pozițiile pe care se găsește valoarea s_1 și $a_{ijk} = 0$ pentru celelalte poziții.

La următorul pas, pentru fiecare element care conține valoarea s_2 vom număra elementele vecine care au valoarea 1 și vom atribui elementului corespunzător valoarea obținută. Practic, se poate spune că vom aduna toate valorile elementelor vecine.

La al treilea pas, pentru fiecare element care conține valoarea s_3 vom calcula valoarea corespunzătoare însumând valorile elementelor vecine care corespund unor cubulețe în care se află litera s_2.

Se observă că, practic, și la al doilea pas se poate realiza o operație similară: pentru toate elementele corespunzătoare unui cubuleț în care se află litera s_2, se însumează valorile elementelor vecine care corespund unor cubulețe care conțin litera s_1.

În general, pentru toate elementele corespunzătoare unui cubuleț în care se află litera s_i, se însumează valorile elementelor vecine care corespund unor cubulețe care conțin litera s_{i-1} (pentru $i > 1$).

Soluția este dată de suma elementelor corespunzătoare cubulețelor în care se află litera s_n, unde n este lungimea cuvântului dat.

Datorită faptului că literele cuvântului sunt distincte, nu apar probleme suplimentare la calcularea valorilor elementelor. Dacă literele s-ar fi putut repeta, în cazul a două litere consecutive identice, ar fi putut apărea erori. Soluția ar fi fost dată de folosirea unui tablou tridimensional suplimentar care ar fi păstrat valorile calculate la iterația anterioară.

Au rezolvat corect:

Mugurel-Ionuț Andreica, București
Adrian Balașko, Zalău
Cristian Băicoianu, Ploiești
Vlad Dascălu, Bacău
Tiberiu Dănet, București
Radu Dondera, București
Octavian-Daniel Dumitran, București
Ștefan Gedo, București
Andrei Giurgiu, București
Andrei Homescu, Târgu-Jiu
Quoc Vinh Ly, Vietnam
Maximilian Machedon, București
Silvestru-Cosmin Negruțeri, Bistrița
Bogdan Nicolae, Sibiu
Sorin Otescu, Brașov
Flaviu Pașca, Bistrița
Csaba Pătcaș, Oradea
Alpár-Ferenc Perini, Sfântu Gheorghe
Mihail-Cosmin Pit-Rada, Dr. Tr. Severin
Constantin Răuțu, Botoșani

Datorită faptului că numărul de apariții ale cuvântului este de cel mult 2.000.000.000, fiecare element al tabloului tridimensional necesită patru bytes pentru a fi reprezentat. Dimensiunea maximă a laturii cubului este 39, deci cantitatea de memorie necesară pentru a păstra elementele este de $39^3 \cdot 4 = 237276$ bytes $\approx$ 232 KB.

Pentru a păstra literele din cubulețe avem nevoie de alți $39^3 = 59319$ bytes $\approx$ 58 KB. Așadar, dimensiunea totală a structurilor de date pe care le folosim este de aproximativ 290 KB, deci se încadrează în limita de 300 KB specificată în regulamentul concursului.

Numărul de parcurgeri ale tabloului tridimensional este egal cu lungimea n a șirului dat. Ordinul de complexitate al unei parcurgeri este $O(k^3)$, unde k este lungimea laturii cubului. Ca urmare, ordinul de complexitate al algoritmului descris este $O(n \cdot k^3)$.

Codul sursă al programului este disponibil pentru *download* la adresa **www.ginfo.ro/concurs/cub.shtml**.

GInfo Top 50

1.	Mugurel-Ionuț Andreica, București	839
2.	Silvestru-Cosmin Negrușeri, Bistrița	790
3.	Adrian-Nicolae Cârcu, Bistrița	714
4.	Adrian Balașko, Zalău	697
5.	Csaba András, Oradea	668
6.	Tiberiu Dăneț, București	657
7.	Octavian-Daniel Dumitran, București	641
8.	Bogdan Nicolae, Sibiu	636
9.	Radu-Andrei Ștefan, Brașov	599
10.	Ștefăniță Fechete, Suceava	597
11.	Andrei Vancea, Cluj-Napoca	575
12.	Maximilian Machedon, București	561
13.	Radu Berinde, București	545
14.	Radu Dondera, București	534
15.	Ioan-Claudiu Gruia, București	521
16.	Csaba Pătcaș, Oradea	518
17.	Claudiu Raicu, Pitești	507
18.	Paul Rusu, Cluj-Napoca	504
19.	Andrei Homescu, Târgu-Jiu	486
20.	Emilian Miron, Bacău	482
21.	Liviu-Cosmin Andreicuț, Bacău	469
22.	Victor-Marius Costan, București	375
23.	Constantin Răuțu, Botoșani	365
24.	Alpár-Ferenc Perini, Sfântu Gheorghe	351
25.	Andrei Giurgiu, București	337
26.	Liviu-Laurențiu Iacob, București	332
27.	Ștefan Gheorghe, București	319
28.	Sorin Otescu, Brașov	310
29.	Flaviu Pașca, Bistrița	307
30.	Mihail-Cosmin Pițu-Rada, Dr. Tr. Severin	288
31.	Andrei Benea, Focșani	280
32.	Vlad Dascălu, Bacău	274
33.	Aurelian Marinescu, Brașov	271
34.	Kovychev Ruslan Alexanderovich, Rusia	268
35.	Dan Ghinea, București	249
36.	Liviu Lalescu, Craiova	245
37.	Mihai-Daniel Cilidariu, Botoșani	244
38.	Mihai Pantelimon, București	243
39.	Adrian Airinei, Bacău	240
40.	Andrei-Vasile Csibi, Cluj-Napoca	232
40.	Bogdan-Vasile Hârjoc, Cugir	232
42.	Vadim Barkalov, S.U.A.	231
43.	Cristian Bojinovici, București	229
44.	Victoria Tarasova, Ucraina	225
45.	Ștefan Bucur, București	218
46.	Cristian Băicoianu, Ploiești	211
46.	Valentin Gușă, București	211
48.	Alexandru Ivan, București	209
49.	Mihail Popa, București	204
50.	Andrei David, Râmnicu Vâlcea	203

L-câmpuri

Lucian Ilea, Andrei Csibi

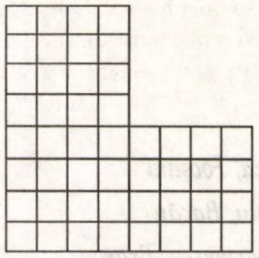
În cadrul acestui articol vom prezenta modul în care o formă geometrică, numită L-câmp, poate fi acoperită folosind forme de același tip, dar de dimensiuni mai mici.

Enunțul problemei

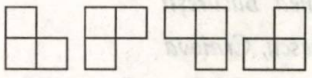
La ediția din anul 1994 a *Olimpiadei de Informatică a Europei Centrale* a fost propusă următoarea problemă:

Descrierea problemei

Definim un *L-câmp* de gradul N o alăturare de trei pătrate, având fiecare dimensiunea 2^N, care sunt dispuse în forma literei *L*. De exemplu, *L-câmpul* de gradul doi este:



Definim *L-forma* ca fiind o structură de trei pătrate de dimensiune 1 × 1, având una dintre următoarele orientări:



Pentru a acoperi un *L-câmp* de gradul N, avem la dispoziție *L-forme* având trei culori, identificate prin valorile 1, 2 și 3. *L-câmpul* trebuie acoperit cu *L-forme*, astfel încât oricare două *L-forme* care au cel puțin o latură comună să fie colorate diferit. *L-formele* de aceeași culoare pot avea orientări diferite dacă se respectă condiția ca două *L-forme* învecinate să fie colorate diferit.

Date de intrare

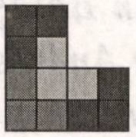
Singura valoare care se va citi de la tastatură este numărul N care indică dimensiunea *L-câmpului*.

Date de ieșire

Pe ecran se va ilustra, folosind trei culori diferite, modul în care poate fi acoperit *L-câmpul* folosind *L-forme*.

Rezolvare

Soluția se poate obține într-o manieră recursivă, dacă se constată descompunerea unui *L-câmp* de ordin N în patru *L-câmpuri* de ordin N - 1 (care pot fi colorate respectând cerințele problemei), așa cum se arată în figura următoare:



Se observă că două dintre cele patru zone vor fi colorate identic; culoarea respectivă este denumită ***culoare majoritară***. Procedeul va continua recursiv până în momentul în care se vor obține *L-câmpuri* de ordin 1.

Din figura anterioară rezultă faptul că *L-câmpurile* de ordin 1 pot fi acoperite în condițiile cerute.

În cazul ordinelor mai mari, *L-câmpurile* obținute după prima descompunere vor trebui, la rândul lor, descompuse. Pentru a nu avea două *L-forme* alăturate care să fie colorate cu aceeași culoare, va trebui să găsim o modalitate de alegere a culorilor. Această modalitate este ilustrată în imaginea alăturată pentru un *L-câmp* de ordin 4. Metoda poate fi aplicată, indiferent de ordinul *L-câmpului*.

Observație

O *L-formă* poate fi considerată ca fiind un *L-câmp* de ordin 0. Așadar, practic, se poate spune că forma inițială a fost definită de forme de același tip, dar de dimensiuni mai mici. Pentru a acoperi un *L-câmp* de ordin n cu *L-câmpuri* de ordin k, va trebui, practic, să acoperim cu *L-forme* un *L-câmp* de ordin $n - k$.

Implementare

Codul sursă al programului care implementează algoritmul prezentat este disponibil pentru *download* la adresa **www.ginfo.ro/revista/11_3/index.shtml**.

Dl. prof. Lucian-Ilea este cadru didactic la Liceul "Emil Racoviță" din Cluj și poate fi contactat prin e-mail la adresa **ilucian@email.ro**. *Andrei Csibi este elev în clasa a XII-a la Liceul "Emil Racoviță" din Cluj și poate fi contactat prin e-mail la adresa* **acsibi@email.ro**.

Despre CONCURSURI

Mihai Stroe

Articolul de față își propune să vină în ajutorul elevilor care doresc să obțină performanțe la concursurile de informatică. Observațiile din cadrul acestui articol sunt rezultatul experienței autorului; ele nu constituie "axiome", ci mai curând sfaturi, valabile atât pentru elevii aflați la început, cât și pentru cei cu o mai mare experiență și participări la concursuri importante.

Concursurile de programare

O primă întrebare la care doresc să afle răspunsul cei care vor să participe la olimpiadele de informatică este: "*Care sunt cele mai importante concursuri de programare*?". În cele ce urmează vom încerca să răspundem la această întrebare.

IOI

Bineînțeles, pentru a răspunde la întrebare trebuie să începem cu *Olimpiada Internațională de Informatică*. Acest concurs reunește anual elevi de liceu din diferite țări ale lumii. Fiecare țară este reprezentată de cel mult patru concurenți, la ultima ediție, cea din 2001 de la *Tampere* (*Finlanda*), au fost reprezentate 75 de țări.

Concursul este individual și se desfășoară sub forma a două probe. La fiecare probă concurenții au la dispoziție cinci ore pentru a rezolva trei probleme cu un grad ridicat de dificultate. După fiecare probă, programele concurenților sunt evaluate automat, cu ajutorul unor programe de evaluare. După cele două zile de concurs se stabilește clasamentul final și se acordă medaliile. Rezultatele sunt secrete până în momentul decernării premiilor. Se stabilesc și clasamente neoficiale pe națiuni (în funcție de puncte și în funcție de medalii).

La aproape toate edițiile acestui concurs, echipa *României* a avut rezultate excelente. Astfel, începând din anul 1993, toți cei patru componenți ai echipei țării noastre au obținut medalii. Datorită faptului că la astfel de concursuri se acordă mai multe premii I, există și noțiunea de "***loc I absolut***"; în acest sens menționăm că *România* a obținut, două locuri I cu punctaj maxim (în 1993 și 1998) și un loc II (în 2001), iar în clasamentul pe națiuni *România* este o prezență constantă în primele cinci locuri.

Mulți se întreabă cum a fost posibilă obținerea acestor rezultate în condițiile în care știm cu toții că situația învățământului românesc este destul de precară. Totul s-a realizat prin munca susținută a elevilor, profesorilor pregătitori, membrilor *Comisiei Naționale* etc. De la an la an, elevilor li se oferă mai multe șanse de afirmare, apar noi concursuri, iar cele vechi sunt mai bine puse la punct, se încearcă adaptarea concursurilor la standardele internaționale etc.

BOI și CEOI

O dovadă a acestei preocupări este inițierea de către țara noastră a două concursuri internaționale regionale. Primul dintre ele este *Olimpiada de Informatică a Europei Centrale* (*CEOI*) a cărui primă ediție a avut loc la *Cluj-Napoca* în anul 1994. Ediția din anul 2000 a acestui concurs a avut loc tot la *Cluj-Napoca*. Standardele acestui concurs sunt mai ridicate decât cele de la *IOI*, deoarece elevii din țările participante sunt întotdeauna foarte bine pregătiți.

Un alt concurs important este *Balcaniada de Informatică* (*BOI*) a cărui primă ediție a avut loc la *Cosntanța* în anul 1993. Standardele acestui concurs nu sunt foarte ridicate, iar concurenții români se clasează întotdeauna pe primele locuri. Cel mai concludent exemplu în acest sens este faptul că la ediția din anul 2001 toate cele patru medalii de aur au fost obținute de reprezentanți ai României.

Concursuri naționale

Pentru a participa la un concurs internațional, un elev trebuie să parcurgă mai multe etape. O parte dintre acestea (cele ***obligatorii***) sunt: participarea la olimpiadele locale și județene, calificarea la *Olimpiada Națională*, obținerea unui rezultat care să permită participarea la barajele pentru selecția în lotul național, obținerea unuia dintre primele 15 locuri la aceste baraje, apoi obținerea unor rezultate bune în cadrul lotului.

Lotul național lărgit (în componența căruia se află aproximativ 15 elevi) participă la mai multe pregătiri, în

cadrul cărora sunt incluse câteva baraje pentru selectarea primilor patru elevi (care vor participa la *IOI*) și formarea echipelor pentru *CEOI* și *BOI*.

Alte concursuri

Există câteva concursuri regionale și naționale similare olimpiadelor (diferă timpul de concurs, numărul de probe și dificultatea problemelor). Câteva dintre acestea sunt:

- *Marele Premiu al Palatului Copiilor* - concurs organizat la *Palatul Național al Copiilor* din *București*; participă echipe ale *Cluburilor Copiilor* din mai multe județe ale țării;
- Concursul "*Grigore C. Moisil*" organizat la *Lugoj* în fiecare an - are o desfășurare similară;
- diferite concursuri interjudețene organizate în anumite regiuni ale țării.

De obicei, aceste manifestări sunt mai ample; la aceste concursuri există și alte secțiuni, cum ar fi cele dedicate dezvoltării de aplicații *software*. Există și alte concursuri destinate exclusiv aplicațiilor *software* (de exemplu concursul organizat de grupul *Microinformatica* de la *Cluj*), care sunt și ele utile în pregătirea pentru olimpiade, deoarece crearea aplicațiilor ajută la dezvoltarea gândirii algoritmice.

Pe plan național au apărut în ultimii ani câteva concursuri la care problemele se rezolvă acasă. În cazul acestora, problemele sunt publicate în anumite reviste și/sau pe *site*-uri *Web*; rezolvările se trimit (de obicei prin *e-mail*) înaintea expirării unui anumit termen. Există câteva runde de acest tip, după care primii clasați se întâlnesc la "***marea finală***", care se desfășoara sub forma unei probe de concurs care este similară cu cea de la olimpiade.

Unul dintre cele mai cunoscute concursuri de acest tip este cel organizat de revista noastră.

Un alt concurs similar este *Cupa Compaq* care este organizat începând cu anul 1995 și constă în două runde care se desfășoară "la distanță" și o etapă finală. Enunțurile problemelor de la primele două runde sunt publicate în revista *PC World*.

Un concurs nou, aflat acum la prima ediție, este *Olimpiada Online*. Organizarea acestuia este similară cu cea a concursului nostru (cinci runde "la distanță" și o etapă finală). Din nefericire, nici una dintre revistele *IT* din *România* nu publică enunțurile problemelor în paginile sale, acestea fiind disponibile doar *online*. *Site*-ul concursului este **www.olimpiadaonline.ro**. Acest concurs are două secțiuni: una dedicată liceenilor și una dedicată elevilor din ciclul gimnazial.

Recomandăm participarea la acest tip de concursuri, atât pentru valoarea premiilor puse în joc, cât și ca modalitate de antrenament.

Ultima categorie de concursuri la care ne oprim sunt concursurile *online* organizate în afara *României* și la care puteți participa prin intermediul *Internet*-ului.

În acest caz, problemele sunt disponibile un anumit interval de timp, în care se pot trimite rezolvările. Aceste concursuri constituie o resursă importantă de probleme pentru pregătire.

Cele mai importante *site*-uri care oferă concursuri *online* sunt **acm.uva.es** și **acm.timus.ru**. Ambele *site*-uri vă pun la dispoziție câte o arhivă foarte bogată de probleme.

Un alt concurs *online* important este *USACO*, prin intermediul căruia se selectează lotul național lărgit al *SUA*. Concursul constă în mai multe faze, desfășurate pe parcursul anului. Prin bunăvoința organizatorilor, concursul a devenit internațional, iar ultimele etape au adus ca noutate traducerea problemelor în mai multe limbi, printre care și *limba română*.

Merită menționat și *Internet Problem Solving Contest*, concurs la care se pun la dispoziție fișierele de intrare și se așteaptă fișiere de ieșire, fără a se cere și programele care rezolvă problemele.

Nu putem acoperi integral în acest articol toate concursurile disponibile pe *Internet*. Există multe alte concursuri; este foarte importantă citirea regulamentului înainte de rezolvarea problemelor, deoarece pot exista restricții care diferă de la caz la caz. Datorită faptului că majoritatea acestor concursuri sunt internaționale, cunoașterea limbii engleze devine o necesitate pentru a putea participa.

Pregătirea pentru concursuri

Probabil, cea mai importantă etapă care trebuie parcursă înaintea prezentării la un concurs este ***pregătirea***, care ar trebui să se desfășoare tot timpul anului, nu numai în perioada premergătoare concursurilor.

Prima observație este aceea că pregătirea trebuie să fie un proces cu o anumită continuitate. O pregătire intensă înaintea unui concurs este foarte importantă deoarece îmbunătățește viteza de implementare și reduce riscul apariției erorilor, dar nu poate rezolva anumite lacune teoretice.

Există mai multe aspecte ale pregătirii:

- pregătirea teoretică;
- simularea unor probe de concurs;
- descuțiile cu alți elevi și profesori referitor la anumite probleme;
- pregătirea psihică;
- pregătirea la locul desfășurării probei.

Pregătirea teoretică

Majoritatea problemelor propuse spre rezolvare la concursuri depășesc nivelul manualelor de informatică. De exemplu, deși se propun o mulțime de probleme a căror rezolvare implică deținerea de cunoștințe din domeniul teoriei grafurilor, nu toți algoritmii necesari sunt cuprinși în programa de învățământ. De aceea, este necesară studierea unor cărți care să acopere un spațiu teoretic cât mai vast.

"Biblia" algoritmilor este considerată așa-numita *CLR*. Denumirea provine de la inițialele numelor celor trei autori (*Cormen*, *Leiserson*, *Rivest*); titlul cărții este *Introducere în algoritmi*. Până acum câțiva ani cartea era foarte greu de găsit și puținii norocoși care reușeau să o obțină aveau un avantaj important la aceste concursuri. Din feri-

cire, între timp a apărut traducerea în limba română a acestei cărți și acum, oricine este interesat, o poate achiziționa.

Câteva dintre cărțile care ar trebui parcurse sunt cele scrise de foști olimpici; prin intermediul acestora, autorii vă împărtășesc o parte din experiența acumulată. Cele mai cunoscute astfel de cărți sunt:

- *Proiectarea și implementarea algoritmilor* - scrisă de *Mihai Oltean* și apărută la editura *Computer Libris Agora* din *Cluj*;
- *Culegere de probleme și programe PASCAL* - scrisă de autorul acestui articol în colaborare cu *Cristian Cadar* și apărută la editura *Petrion* din *București*;
- *Psihologia concursurilor de informatică* - scrisă de *Cătălin Frâncu* și apărută la editura *L&S* din *București*.

Simularea unor probe de concurs

Cea mai potrivită modalitate de pregătire pentru a face față unei situații este simularea ei, adică tratarea unei situații asemănătoare. De exemplu, un fotbalist care execută foarte bine loviturile libere la antrenamente, are șanse mari să le execute la fel de bine și în timpul meciului deoarece este bine pregătit pentru această situație. Același principiu se aplică și în domeniul concursurilor de programare. Dacă vă pregătiți pentru un concurs, citirea unor cărți nu este suficientă! Trebuie să vă analizați comportamentul în situații similare unui concurs real. Situația cea mai asemănătoare unui anumit concurs este un alt concurs de același tip! Participarea la N concursuri crește șansele obținerii unui rezultat mai bun la al N+1-lea.

Concursurile de pe *Internet* sunt destul de dese și sunt organizate foarte bine. Din nefericire, nivelul de dificultate nu este întotdeauna cel dorit de cel care se pregătește.

O posibilitate de simulare a unui concurs este rezolvarea problemelor de la o ediție precedentă! De exemplu, cineva care se pregătește pentru *Olimpiada Națională* din 2002 ar trebui să rezolve problemele date la *Olimpiada Națională* din 2001 la clasa respectivă (există și cazuri în care simulări de acest tip nu sunt foarte concludente; de exemplu, între 2000 și 2001 programa școlară a suferit câteva variații care s-au reflectat în tipul problemelor de la clasa a X-a).

Pentru ca simularea să reflecte cât mai bine realitatea, problemele se rezolvă în timpul stabilit, fără pauze, la prima citire (sau, dacă au fost citite anterior, nu se recitesc în zilele premergătoare simulării).

Experiența obținută este apropiată de cea a concursului propriu-zis, dar lipsește *stress*-ul care apare, inevitabil, în timpul competiției.

Este important ca, după fiecare simulare sau concurs real, să vă analizați comportarea și să învățați din eventualele greșeli de abordare (de exemplu, puteți ajunge la concluzii de genul "*Această problemă trebuia abordată prima*" sau "*Nu am citit integral enunțul și am rezolvat o altă problemă*"). În plus, se recomandă notarea celor mai frecvente greșeli de implementare și examinarea periodică a listei; în acest fel, ele vor dispărea ulterior.

Discuțiile cu alți elevi și profesori

Puteți învăța foarte mult de la profesori sau elevi mai experimentați! Există exemple de elevi pentru care a contat foarte mult pregătirea individuală, dar majoritatea au fost ajutați de pregătirea organizată, în grupuri de elevi, sub îndrumarea unor profesori cu preocupări de acest gen. În cadrul pregătirilor de acest tip se discută algoritmi, se propun probleme spre rezolvare, se discută diversele modalități de rezolvare, se obțin mai multe informații despre concursuri etc. În plus, elevii aduc în discuție diverse probleme cu care s-au întâlnit în cadrul pregătirii individuale.

Dacă doriți să participați la astfel de pregătiri, trebuie să luați legătura cu alți elevi interesați de concursuri, sau cu profesorii care se ocupă de olimpiade. În prezent se organizează pregătiri la nivel de liceu, oraș, județ etc. Astfel de pregătiri cresc valoarea tuturor participanților, deci sunt foarte importante.

Pregătirea psihică

Nu vom face aici un tratat de psihologie. În schimb vom formula câteva observații de bun-simț și vom desființa anumite prejudecăți.

Este cunoscut faptul că o atitudine mentală pozitivă este cheia succesului în cele mai multe situații. Din nefericire, unii concurenți încep proba cu un moral nu tocmai ridicat.

Iată câteva din falsele probleme cu care se confruntă anumiți concurenți:

- *Participă și X, care e mai bun ca mine, deci nu am nici o șansă să câștig!*

  Fals! Nu s-a demonstrat că X este mai bun, cel mult a obținut rezultate mai bune până acum și poate avea șanse mai mari. Problemele din concursul curent sunt aceleași pentru toți, condițiile de desfășurare sunt aceleași și antecedentele nu contează. Totul se reia de la zero.
- *Am obținut prea puține puncte în prima zi, nu mai am nici o șansă la premii!*

  Problemele de la concursurile cu mai multe probe sunt, în general, destul de dificile. De multe ori, la *Olimpiada Națională* sau la concursurile internaționale, obținerea a jumătate din punctele puse în joc înseamnă câștigarea unui premiu. Dacă în prima zi rezultatele obținute sunt nesatisfăcătoare, un rezultat foarte bun în ziua a doua poate aduce premiul dorit.
- *Comisia nu este imparțială!*

  Deși foarte rar, unii elevi susțin acest lucru. Mulți alții nu au curajul să afirme așa ceva, dar sunt convinși că nu au aceleași șanse ca și preferații comisiei. Puteți fi siguri că în cazul concursurilor de informatică (cele care se respectă) comisia este imparțială și subiectivismul este eliminat datorită evaluării automate. Dacă punctajul nu pare a fi în concordanță cu cel așteptat, depuneți contestații și cereți lămuriri! Dacă aveți dreptate, veți primi punctajul corect. Singurul caz în care este riscant să cereți o reevaluare apare atunci când rezolvarea voastră folosește numere generate aleator.

• *Nu sunt suficient de bine pregătit!*

Această apreciere este mai realistă. Totuși, trebuie să știți că o foarte mare importanță o are inspirația de moment sau norocul (poate problemele vor fi similare cu unele rezolvate anterior).

Evident, optimismul exagerat poate, la rândul său, să fie dăunător. Cel mai bine ar fi să adoptați atitudinea cea mai potrivită pentru propria personalitate. Veți observa, în timp, care este aceasta.

Pregătirea la locul desfășurării probei

Deși ar putea părea bizar, acest aspect este foarte important, dar este neglijat aproape întotdeauna de către concurenți. Această pregătire constă în:

• somn odihnitor în noaptea care precede ziua concursului;
• obținerea atitudinii mentale dorite;
• prezentarea la timp în sală, cu toate obiectele necesare.

Există o mulțime de obiecte pe care ar trebui să le aveți la voi în timpul desfășurării probei. Vom încerca să le amintim pe cele mai importante.

În primul rând trebuie să aveți un ceas pentru a ști întotdeauna cât timp mai aveți la dispoziție. Nu este bine să vă bazați doar pe ceasul calculatorului, deoarece consultarea acestuia necesită timp care nu mai poate fi folosit pentru rezolvarea problemelor.

Trebuie să aveți instrumente de scris, foi albe și foi cu pătrățele (pentru a rezolva problemele care necesită cunoștințe de geometrie analitică). De obicei, organizatorii vă pun la dispoziție foi albe, dar este bine să nu riscați.

Nu în ultimul rând, ar fi bine să aveți o sticlă de suc și o ciocolată sau un *croissant*; din nefericire, concursul începe deseori cu întârziere și este bine ca foamea sau setea să nu vă preocupe în timpul rezolvării problemelor.

Concursul propriu-zis

De acum vom presupune că sunteți așezat în fața calculatorului și primiți problemele. Ne vom concentra, în principal, asupra concursurilor de tipul olimpiadelor naționale și internaționale; totuși, majoritatea recomandărilor sunt valabile și pentru alte tipuri de concursuri.

De obicei, veți primi problemele listate pe foi. Pentru început va trebui să verificați dacă ați primit toate problemele! În situația puțin probabilă în care lipsesc anumite foi, trebuie să anunțați imediat acest lucru.

După primirea subiectelor, veți începe lecturarea lor. La începutul probei trebuie să citiți integral toate problemele pentru a vă forma o imagine de ansamblu asupra acestora și pentru a descoperi eventuale ambiguități. Problemele se citesc integral, toate la începutul probei, pentru a se obține o imagine de ansamblu și pentru a se evidenția eventualele ambiguități. Nu trebuie să începeți implementarea imediat ce aveți o idee de rezolvare pentru una dintre probleme.

Este foarte important să citiți toate enunțurile. Altfel spus, în primele zece minute (sau mai mult) nu se atinge calculatorul! Întotdeauna, când citiți o problemă, este indicat să întoarceți foaia pentru a vedea dacă enunțul continuă și pe verso!

De obicei, în primele 30 sau 60 de minute ale concursului se pot adresa întrebări comisiei, pentru a clarifica eventualele ambiguități din enunțuri. Acestea sunt redactate în scris, foile sunt preluate de supraveghetorul din sală și trimise la comisie.

Răspunsul s-ar putea să întârzie, deci este indicat să nu irosiți timpul așteptându-l fără a mai face nimic altceva. Puteți fie să vă gândiți la rezolvarea unei probleme, fie să începeți să implementați secvențe simple, cum ar fi rutinele pentru citirea datelor de intrare.

În majoritatea situațiilor, întrebările trebuie formulate în așa fel încât să se poată răspunde prin ***Da*** sau ***Nu***. Dacă întrebarea nu este astfel exprimată sau dacă răspunsul acesteia se găsește în textul problemei respective, veți primi răspunsul ***Fără comentarii***, caz în care va trebui mai întâi să studiați corectitudinea întrebării și, dacă aceasta este corect formulată, să recitiți enunțul problemei.

Concurenții trebuie să profite cât mai mult de această perioadă, pentru a clarifica eventualele nelămuriri. Pentru a rezolva problemele trebuie, în primul rând, să știți care sunt cerințele!

Este foarte important ca la întrebări să răspundă membrii comisiei. Au existat unele cazuri în care supraveghetorii își dădeau cu părerea asupra enunțului; dacă aceștia nu fac parte din comisie, răspunsul s-ar putea să nu fie cel corect. Nu trebuie să acceptați răspunsuri dacă acestea nu sunt însoțite de semnătura unui membru al comisiei.

După ce toate problemele sunt clare, se trece la căutarea algoritmilor de rezolvare. Și în această etapă, se vor aborda toate problemele, dar analiza va fi rapidă. Problemele pentru care nu se găsește algoritmul de rezolvare vor fi abordate din nou mai târziu.

Enunțul problemei oferă și indicații asupra ordinului de complexitate al algoritmului de rezolvare prin limitele asociate datelor de intrare și precizarea timpului de execuție. Dacă aceste informații nu sunt specificate, va trebui neapărat să semnalați acest lucru prin întrebări adresate comisiei.

Dimensiunea datelor de intrare oferă indicații asupra complexității rezolvării așteptate. De exemplu, dacă datele de intrare constau într-un graf cu 1000 de noduri și 5000 de muchii, se așteaptă un algoritm cu ordinul de complexitate cel mult pătratic. Este mult mai probabil ca algoritmul căutat să fie liniar sau liniar-logaritmic.

După această fază, aveți deja o primă idee asupra:

• problemelor și algoritmilor de rezolvare;
• punctajului așteptat (de exemplu, un *backtracking* pentru $N = 1000$ vă va aduce foarte puține puncte);
• timpului necesar pentru implementarea algoritmilor de rezolvare pentru anumite probleme.

Dacă există o problemă simplă, care se poate rezolva foarte repede, este bine să începeți cu aceasta!

Dacă ați rezolvat repede cel puțin o problemă, vă rămâne destul de mult timp pentru restul concursului (de

multe ori rezolvarea corectă a două probleme din trei înseamnă obținerea unui rezultat excelent).

După rezolvarea problemelor simple, se analizează restul problemelor. De obicei, în această fază există probleme pentru care concurentul nu are nici o idee de rezolvare (unele dintre acestea pot deveni simple printr-o analiză mai atentă), probleme a căror implementare necesită mult timp și probleme pentru care idea de rezolvare curentă nu va aduce punctajul maxim.

Indiferent ce problemă decideți să abordați în continuare, țineți cont de următoarele observații:

- scopul concurentului nu este să rezolve corect problemele, ci să obțină cât mai multe puncte; este mult mai bine să implementați în 10-15 minute o rezolvare care poate aduce 10-20% din punctajul maxim, decât să nu rezolvați deloc problema;
- programele pe care le veți preda comisiei trebuie să realizeze exact ceea ce se cere în specificațiile problemei.

După realizarea unui program care rezolvă o anumită problemă, se impune testarea acestuia. Este momentul unei recitiri a enunțului problemei (dacă au existat unele scăpări, acum este timpul să fie înlăturate).

Fiecare program este testat pentru testul simplu din exemplul dat în enunț și pentru cât mai multe teste elaborate manual.

Dacă aveți timp, puteți crea un generator de teste și un evaluator care verifică anumite caracteristici ale soluției obținute de programul vostru.

De exemplu, dacă într-o problemă se cere un drum minim într-un graf, generatorul va genera grafuri aleatoare sau cu anumite caracteristici, iar evaluatorul ar putea verifica dacă muchiile afișate se găsesc în graf, lungimea drumului este cea declarată etc.

Majoritatea concurenților nu vor scrie un generator și un evaluator pentru această problemă; totuși, pentru probleme mai complicate, aceste programe auxiliare sunt de un real ajutor. Astfel, se pot depista unele greșeli de implementare și se estimează timpul de execuție pentru date de intrare mari. Ratarea unei probleme din cauza unei greșeli de implementare este foarte neplăcută; concurentul pierde puncte importante, pe care le-ar fi putut obține și pe care ceilalți, probabil, le-au obținut.

Un program care nu se încadrează în timp va trebui optimizat. Se recomandă să păstrați o versiune funcțională a acestuia (dacă nu aveți timp să terminați sau greșiți la implementare și nu mai puteți reveni la forma anterioară, ați pierdut foarte mult timp și nu este deloc convenabil să scrieți din nou anumite secvențe din program).

În cazul în care nu știți să rezolvați perfect o problemă, dar ea poate fi abordată prin *backtracking*, cu șanse de obținere a unui punctaj mulțumitor, nu uitați de celelalte alternative: metode euristice *greedy*, algoritmi probabiliști etc. Eventual, puteți combina aceste metode (de exemplu, puteți folosi un *greedy* și apoi, până la expirarea timpului, puteți încerca să îmbunătățiți prin *backtracking* soluția obținută).

După ce ați terminat problemele (se întâmplă destul de rar să reușiți) nu ieșiți din sală! Este momentul ultimelor teste. La ieșirea din sală trebuie să fiți convinși că ați făcut tot ce era posibil în condițiile date.

Concursul nu s-a terminat încă! Urmează corectarea! Va trebui să verificați punctajul obținut și să fiți pregătit să depuneți o contestație dacă aveți impresia că ceva nu este în regulă.

La unele concursuri, corectarea se face în prezența concurentului; aici aveți ocazia să solicitați să vi se arate testele și ieșirile furnizate de programul vostru, să cereți testarea din afara mediului de evaluare etc.

La alte concursuri, comisia oferă, mai târziu, testele și răspunsurile corecte pentru autoevaluare. Nu ratați ocazia de a vă evalua rezolvările și nu depuneți contestații decât dacă în urma autoevaluării obțineți un punctaj mai mare.

Nu are sens să depuneți contestații inutile. Nu are nici un rost să creați neplăceri comisiei. Trebuie să fiți capabili să vă dați seama că ați greșit. Nu trebuie să aveți pretenții absurde! S-ar putea ca în viitor să vă mai întâlniți cu membrii comisiei respective și la alte concursuri și animozitățile create nu pot decât să vă dezavantajeze. Totuși, dacă sunteți sigur că aveți dreptate, cereți lămuriri până în momentul în care vi se acceptă contestația sau vi se dovedește că ați greșit.

Fiecare concurs este o experiență în plus. Încercați să învățați din greșeli. Discutați după probă cu alți concurenți, aflați cum ar fi trebuit rezolvate problemele pe care nu le-ați știut aborda și ce au greșit ceilalți (este bine să învățați și din greșelile altora).

Dacă primele rezultate nu sunt extraordinare, nu trebuie să abandonați! Poate a fost o zi proastă, sau poate ceilalți au mai multă experiență. Situația se va ameliora în timp!

Încheiere

Poate este puțin cam târziu dar, vom încerca să răspundem și la următoarea întrebare: *De ce să particip la astfel de concursuri?*

Principalele avantaje sunt formarea unei gândiri algoritmice, înțelegerea metodelor de rezolvare pentru anumite probleme (care pot apărea și în cadrul dezvoltării de aplicații), dezvoltarea capacității de a reacționa rapid într-un timp scurt.

Un avantaj deloc de neglijat este reprezentat de premiile puse în joc, atât cele materiale (calculatoare, diferite componente *hardware*, excursii în străinătate, cărți etc.), cât și intrarea fără examen la facultate, în cazul obținerii unor rezultate bune la olimpiadele naționale sau internaționale.

Încheiem prin a vă ura succes la concursurile la care veți participa. Sperăm că aceste recomandări vă vor fi de un real folos!

Mihai Stroe este student în anul IV la Universitatea Politehnica din București. Poate fi contactat prin e-mail la adresa mstroe@pcnet.pcnet.ro.

Dispozitive MECANICE de calcul

Laviniu-Aurelian Bădulescu

Aparent un domeniu în care totul se conjugă la timpul prezent, calculatoarele au o istorie a lor, care începe cu mii de ani în urmă. În cadrul acestui articol vom prezenta câteva repere istorice importante pentru acest domeniu.

Cel mai îndepărtat strămoș al calculatorului de astăzi este ***abacul***, un instrument de calculat, cu ajutorul unor bile mobile așezate pe bare orizontale și paralele fixate într-un cadru de lemn; el permitea utilizatorilor lui efectuarea unor operații simple de adunare, scădere sau înmulțire.

Probabil, abacul a fost inventat de sumerieni în vechea *Mesopotamie* (*Babilon*), de unde a fost preluat în *Egipt*, *China* și pe valea fluviului *Indus*, înainte de a ajunge în lumea greco-romană. *Grecii* sunt cei care i-au dat numele pe care îl utilizăm astăzi; semnficația cuvântului este ***planșetă*** sau ***tabelă de calcul.***

Abacul a fost în utilizat continuu în *Europa* până în *Evul Mediu*, deci aproape 2000 de ani, fiind treptat eliminat odată cu adoptarea sistemului de numerație indiano-arab care folosea și cifra zero, ceea ce ușura foarte mult calculele.

Totuși, acest instrument este încă utilizat în unele țări asiatice, în special în *China*, de obicei pentru calculele de afaceri.

Arabii au efectuat cercetări intense în domeniul matematicii și, în secolul al *IX*-lea, ***al-Khwarizmi*** a publicat o lucrare fundamentală (***Kitab al-jabr-wa al-muqabalah***) despre un sistem algebric de manipulare a expresiilor numerice care fundamentează bazele calculului zecimal. Cuvântul ***algebră*** provine din denumirea acestei lucrări.

El este cel care a definit pentru prima dată noțiunea de ***algoritm***, cuvânt care este derivat din numele său.

Prin intermediul lucrărilor sale, europenii vor cunoaște cifrele indiene (numite astăzi cifre arabe) și numărul ***zero.***

Nu se poate spune cu exactitate când s-a introdus pentru prima dată cifra zero într-un sistem de numerație. Unii consideră că acest concept era cunoscut vechilor babilonieni, dar este mai probabil ca el să fi apărut în jurul anului 700, în *India*. Cultura *Maya*, independent de indieni și posibil înaintea lor, a dezvoltat un sistem de numerație care includea cifra zero.

Numărul zero este introdus în *Europa* de *Papa Silvestru al* II-*lea* (999-1003). Acest număr este considerat a fi una dintre cele mai ingenioase invenții ale intelectului uman. Banalele calcule aritmetice de adunare sau scădere devin mai dificile dacă nu utilizăm cifra zero. Această cifră face posibilă existența sistemului de numerație zecimal (ne permite să facem diferență între numere cum ar fi 23, 203, 2030 etc.)

Dacă cifra zero nu ar fi existat, atunci nu ar fi putut fi create (sau principiile care ar fi stat la baza lor ar fi fost mult diferite) calculatoarele numerice moderne care lucrează în baza 2, adică în sistemul binar care mai conține în afara cifrei zero doar cifra unu.

Cel care introduce sistemul zecimal în *Europa* este matematicianul italian *Fibonacci.* După studii cu un profesor arab în nordul *Africii*, el a călătorit în *Egipt*, *Siria*, *Grecia*, *Sicilia* și *Proveța*, studiind diverse sisteme de numerație și metode de calcul.

Lucrarea sa fundamentală este ***Liber Abaci***, a fost publicată în anul 1201 și a furnizat bazele pentru scrierile de aritmetică și algebră în secolele următoare. Până la generalizarea folosirii sistemului de numerație zecimal în întreaga *Europă* au mai trecut patru secole.

După abac, alte dispozitive de calcul apar tot în *Orient*, fiind în general legate de observațiile astronomice. Astfel, ***Jamsid al Kashi*** a construit efectiv și a utilizat (în jurul anului 1400) primele dispozitive de calcul specializate pentru observatorul astronomic de la *Samarkand.*

Leonardo Fibonacci

Matematician italian născut la Pisa în jurul anului 1170, a rămas celebru pentru seria numerică ce îi poartă numele, în care fiecare element este suma ultimelor două elemente care îl preced. Chiar dacă seria fost descoperită în secolul al XIII-lea, ea este încă folosită de calculatoarele electronice moderne în cadrul unor algoritmi de sortare a datelor.

Calculatoarele au evoluat din aceste dispozitive de calcul în două direcții: dispozitivele *analogice* și dispozitivele *numerice*, în funcție de tipul mărimilor cu care lucrau (analogice, numite și *continue*, sau numerice, numite și *discrete* sau *digitale*).

În *Europa*, istoria dispozitivelor de calcul evidențiază momentul apariției riglei de calcul, primul calculator analogic remarcabil. Inventatorul ei a fost matematicianul englez ***Edmund Gunter***, care construiește o primă variantă de riglă de calcul în 1620, pentru ca în 1622 să prezinte o variantă îmbunătățită a acesteia.

Rigla de calcul a devenit imediat un instrument foarte utilizat în calculele inginerești și în cele de afaceri și a rămas în uz până la introducerea calculatorului, în ultima jumătate a secolului al *XX*-lea.

Această invenție a fost posibilă datorită descoperirii logaritmilor (1614) de către matematicienii: *John Napier* (scoțian) și *Henry Briggs* (englez). Rigla de calcul a fost utilizată inițial pentru operațiile de înmulțire sau împărțire a numerelor zecimale (prin adunarea sau scăderea logaritmilor acelor numere), iar ulterior a fost perfecționată astfel încât să realizeze și operații de extrageri de radicali, calculul unor funcții trigonometrice sau a logaritmilor.

Anul 1642 marchează realizarea primei mașini mecanice de calcul numeric - al cărei principiu s-a menținut sute de ani - precursoare a calculatorului modern. Este vorba despre mașina mecanică cu roți dințate a lui ***Blaise Pascal***.

Principiul mașinii era simplu: în momentul în care roata unităților realiza o rotație completă (erau parcurși zece dinți), roata zecilor se deplasa cu un dinte (se incrementa cifra zecilor cu o unitate); când roata zecilor realiza o rotație completă, roata sutelor se deplasa cu un dinte etc.

Idea lui *Pascal* va fi îmbunătățită ulterior de alți oameni de știință sau inventatori cum ar fi *Samuel Morland* (1666), *Gottfried Leibniz* (1671), *W.T. Odhner* (1878), *James Ritty* (1879), *Dorr E. Felt* (1886), *Wiliam S. Burroughs* (1891), *Frank S. Baldwin* sau *Jay R. Monroe* (1912).

În 1666, *Samuel Morland* adaptează ideile lui *Pascal* pentru o mașină de înmulțire zecimală bazată tot pe roți dințate. În 1671, *Gottfried Wilhelm Leibniz* construiește o mașină pentru adunare, scădere, înmulțire și împărțire, bazată pe idea lui *Pascal*. El folosea un ax cu dinți fixați în lungul lui și o roată dințată cu zece dinți. Marginea acestei roți dințate era marcată cu cifre de la 0 la 9. Când numărul 4, de exemplu, era introdus în mașină, axul se mișca astfel încât numai 4 din dinții roții să fie cuplați. Când axul era răsucit o dată, roata dințată avansa cu 4 din dinții săi astfel încât numărul indicat pe cadran creștea cu 4 unități; astfel era realizată o adunare. Pentru a înmulți pe 4 cu 5, numărul 4 era introdus în mașină și axul era rotit de 5 ori. Scăderile se făceau rotind în sens invers axul, iar împărțirile erau realizate prin scăderi repetate.

Idea lui *Pascal* era încă aplicată 250 de ani mai târziu; astfel, în 1928, inginerul electrotehnist american ***Vannevar Bush*** (1890-1974) a proiectat și apoi a realizat *analizorul diferențial* (*differential analyzer*), primul calculator mecanic capabil să rezolve ecuații diferențiale. Mașina sa realiza calculele cu numere zecimale și de aceea avea nevoie de sute de angrenaje și axe pentru a reprezenta diferitele mișcări și relații dintre cele zece cifre.

În anul 1801, ***Joseph-Marie Jacquard*** descoperă principiul cartelelor de carton perforate ca purtătoare de informație. El inventează o mașină de țesut care folosea cartelele perforate pentru a obține automat modelul dorit de țesătură. Inventatorul englez ***Charles Babbage*** va folosi cartelele ca un mecanism de comandă al mașinii sale analitice, ***Ada Augusta of Lovelace***, utilizând aceleași cartele, va crea primul program de calculator, iar pentru a culege datele la mașina sa de tabulat (1889) le va folosi și ***Herman Hollerith*** (1860-1929).

Herman Hollerith era un statistician american, angajat la *United States Bureau of the Census* (*Biroul de Recensăminte al SUA*), care de-a lungul deceniului nouă al secolului al *XIX*-lea a studiat modalitatea de a crea o mașină de calcul care să numere, să colecționeze și să sorteze informațiile stocate pe cartele perforate. Principiul de funcționare al mașinii de tabulat era simplu: o cartelă avea mai multe perforații; o perforație pe cartelă, după poziția în care era realizată, codifica o anumită cifră sau literă. Când cartelele erau introduse în mașină, ele erau apăsate de o serie de ace metalice corespunzând unei rețele de perforații potențiale. Când un ac găsea o perforație (perforație care putea reprezenta vârsta, ocupația ș.a.) se închidea un circuit electric și se incrementa valoarea corespunzătoare acelei categorii. Mașina de tabulat a fost utilizată pentru prima dată pentru sortarea datelor statistice culese în timpul recensământului din anul 1890 din *Statele Unite*.

Blaise Pascal (1623-1662)

Unul dintre cei mai străluciți oameni ai timpului său, Pascal a avut contribuții de valoare universală în domeniile științelor, matematicii, religiei și filosofiei. Scrierile sale Les Provinciales (Scrisori provinciale) și Pensées (Gânduri) sunt considerate capodopere ale genului.

Tatăl său s-a ocupat personal de educația lui Pascal, care s-a dovedit a fi un copil precoce. La 12 ani redescoperă axiomele geometriei lui Euclid; la 17 ani scrie un eseu asupra secțiunilor conice, fiind un pionier în acest domeniu al geometriei proiective, unde o teoremă îi poartă numele.

Între 1642 și 1644 construiește prima mașină de calcul numeric pentru a-și ajuta tatăl în activitatea contabilă. Pascal este autorul a numeroase studii și descoperiri în geometrie, hidrodinamică, hidrostatică, presiunea atmosferică, probabilități, aritmetica numerelor, calculul integral etc.

Wilhelm Leibniz (1646-1716)

Deşi nu a fost un artist, Leibniz a fost comparat cu Leonardo da Vinci. El a fost recunoscut ca un geniu universal al timpului său, un om de ştiinţă şi filosof care a acoperit domeniile matematicii, geologiei, teologiei, mecanicii, istoriei, jurisprudenţei şi lingvisticii. Leibniz a urmat cursurile Universităţii din Leipzig şi a obţinut doctoratul în drept la Universitatea din Nurenberg. În timpul vieţii sale, Leibniz a perfecţionat maşina de calculat inventată de Pascal, a pus bazele calculului integral şi diferenţial, a descoperit dinamica, o ramură a mecanicii, a lucrat la diverse dispozitive mecanice (ceasuri, prese hidraulice, lămpi, submarine şi mori de vânt), a perfecţionat sistemul binar de numeraţie utilizat de calculatoarele actuale, a conceput teoria conform căreia întreg raţionamentul poate fi redus la o combinaţie ordonată de elemente ca numere, cuvinte, sunete sau culori (baza teoretică a calculatoarelor moderne), a încercat să creeze o logică bazată pe utilizarea simbolurilor, fiind un precursor al logicii matematice, a pus fundamentele topologiei generale, o ramură a matematicii, s-a străduit să alcătuiască o bază pentru unificarea Bisericii şi a încercat să redacteze o istorie universală. Totodată, a perfecţionat continuu sistemul său metafizic de cercetare a noţiunii de cauză universală a tot ceea ce există, sistemul său filosofic considerând ca principiu al existenţei "monadele", substanţe spirituale indivizibile, independente unele de altele, înzestrate cu o forţă activă. Materia nu este altceva decât manifestarea exterioară a monadelor; fiecare monadă oglindeşte întregul univers; concordanţa dintre activitatea monadelor este asigurată de "armonia prestabilită", creată de monada supremă, Dumnezeu.

Maşina de tabulat a lui *Hollerith* şi-a dovedit imediat superioritatea în raport cu dispozitivele mecanice de manipulare a datelor. Recensământul anterior necesitase şapte ani şi jumătate de muncă pentru a prelucra manual datele culese; utilizând tabulatorul lui *Hollerith* - pentru recensământul din 1890 - au fost necesare şase săptămâni pentru a încărca datele culese, iar analiza şi prelucrarea completă a acestora a necesitat doar doi ani şi jumătate.

În 1896, *Hollerith* a fondat *Tabulation Machine Company*, pentru a produce maşini de tabulat. În 1924, după un număr de fuziuni, compania şi-a schimbat numele în *International Business Machines Corporation* (*IBM*) care este, în momentul de faţă, a patra corporaţie industrială, ca mărime, din *SUA*. *IBM* reuşeşte să impună maşina cu cartele perforate pe piaţă şi producerea acesteia devine cea mai mare afacere cu sisteme informatice până la sfârşitul anilor '60 ai secolului al *XX*-lea, când o nouă generaţie de calculatoare au scos de pe piaţă maşinile care utilizau cartele perforate pentru a introduce datele şi programele în calculator.

Între anii 1914 şi 1952, când firma a fost condusă de *Thomas J. Watson*, *IBM* domină pieţele la capitolul maşini cu cartele perforate şi ceasuri; în anii '30 realizează maşina de scris electrică; în anii '50 se lansează în producţia de calculatoare *mainframe*; la începtul anilor '80 este lansat *IBM PC* (computerul personal). Mai multe detalii despre *IBM* puteţi citi în articolul dedicat acestei companii.

În secolul al *XIX*-lea se dezvoltă şi cercetările teoretice în domeniu, datorită aportului a doi matematicieni englezi: *Morgan* şi *George Boole*. Demonstrând principiile logice cu simboluri matematice în loc de cuvinte, *George Boole* a pus bazele logicii simbolice, un domeniu al studiilor matematico-filosofice.

În noua disciplină, căreia i-a pus bazele, cunoscută ca ***algebră booleană***, toate obiectele sunt divizate în două clase separate, fiecare cu o proprietate dată; fiecare clasă poate fi descrisă în termenii prezenţei sau absenţei aceleiaşi proprietăţi. De exemplu, un circuit electric poate fi sau nu sub tensiune.

Algebra booleană a fost aplicată în proiectarea circuitelor binare ale calculatoarelor şi în echipamentele de comutare din telefonie. Aceste dispozitive sunt sisteme bazate pe două valori booleene (prezenţa sau absenţa proprietăţii). Articolul său *Analiza matematică a logicii* (*Mathematical Analysis of Logic*), publicat în 1847, conţinea prima definire a principiilor logicii simbolice.

Ulterior, *Boole* publică în 1854 *Legile Gândirii* în care fundamentează o nouă logică (*logica bivalentă*), bazată numai pe două valori de adevăr (*fals* şi *adevărat*) care stă la baza întregului aparat matematic folosit în proiectarea logică.

George Boole (1815-1864)

Deşi filosofii studiaseră logica pentru a baza pe raţionamente solide afirmaţiile lor, matematicianul englez George Boole susţine însă că logica trebuie pusă în legătură mai degrabă cu matematica decât cu filosofia. Boole a fost un autodidact; a început să predea de la vârsta de 16 ani pentru a-şi ajuta familia. În timpul său liber citea publicaţii de matematică şi în curând a început să scrie articole pentru acestea. La vârsta de 29 de ani, Boole a primit medalia de aur pentru opera sa de la Societatea Regală Britanică. Doi ani mai târziu a fost numit profesor de matematică la "Queen's College" în Irlanda, cu toate că nu a studiat niciodată la o universitate.

Dl. prof. Laviniu-Aurelian Bădulescu este cadru didactic la Colegiul Naţional "Elena Cuza" din Craiova şi poate fi contactat la adresa **badcri@yahoo.com**.

International Business Machines

Carmen Şuşman

Compania IBM (International Business Machines) este una dintre cele mai cunoscute firme care își desfășoară activitatea în domeniul tehnologiei informației. În cadrul acestui articol veți putea citi câteva informații referitoare la istoria de peste un secol a acestei companii.

Nașterea companiei

Compania *IBM* a luat ființă în anul 1911, sub numele de *C-T-R* (*Computing-Tabulating-Recording Company*). Cu toate acestea, originile sale datează de la sfârșitul secolului al XIX-lea.

Compania *C-T-R* s-a născut prin fuziunea a trei companii, și anume *Tabulating Machine Company*, *Computing Scale Company of America* și *International Time Recording Company*.

Tabulating Machine Company a fost înființată în anul 1896 și a fost condusă de ***Herman Hollerith***. Acesta a construit o mașină de tabulare care folosea cartele perforate, care a fost utilizată de o agenție guvernamentală pentru a realiza statistici referitoare la numărul imigranților sosiți în timpul revoluției industriale.

International Time Recording Company a fost fondată în anul 1900. Compania a construit o fabrică foarte modernă la *Endicott, New York*; ulterior, această fabrică a devenit *IBM Plant No.* 1, una dintre cele mai importante fabrici ale companiei.

Computing Scale Company of America, cea de-a treia companie care a participat la fuziune, a fost fondată în anul 1901.

Inițial, compania *C-T-R* a fost condusă de ***Charles F. Flint***, cel care a organizat fuziunea celor trei companii predecesoare. Pe măsură ce domeniul de activitate al noii companii s-a diversificat, *Flint* și-a dat seama că aceasta devine tot mai greu de condus. El a apelat la ajutorul lui ***Thomas J. Watson*** care a primit funcția de director general al companiei.

Acesta a pus în practică o serie de tactici care au avut un succes neașteptat. După 11 luni *Watson* a devenit președintele companiei. În primii săi patru ani de activitate, veniturile companiei s-au dublat. *C-T-R* și-a extins activitatea și pe alte continente: *Europa*, *America de Sud*, *Asia* și *Australia*. Compania, al cărei sediu central era în *New York*, a ajuns să aibă 1300 de angajați și birouri sau fabrici în diverse orașe cum ar fi *Endicott*, *Binghamton*, *Dayton*, *Ohio*, *Detroit*, *Michigan*, *Washington*, *Toronto* sau *Ontario*.

Gama de produse oferite de *C-T-R* era foarte diversă; compania oferea atât tabulatoare cu cartele perforate, cât și dispozitive pentru tăierea mezelurilor sau a brânzeturilor.

Schimbarea numelui

După terminarea primului război mondial, inginerii și cercetătorii de la *C-T-R* au dezvoltat numeroase produse pentru a veni în întâmpinarea nevoilor crescânde ale clienților.

În anul 1920 a fost lansată pe piață o mașină electrică pentru contabilitate, precum și un sistem complet pentru controlul timpului destinat unităților de învățământ.

În anul 1921 a fost preluată compania *Ticketograph* din *Chicago* și au fost cumpărate mai multe brevete ale companiei *Peirce Accounting Machine*. Ca urmare, au fost dezvoltate produsele care foloseau cartele perforate, performanțele acestora dublându-se până în anul 1924.

Creșterea neașteptată a companiei a făcut ca numele acesteia să nu mai fie de actualitate. Astfel, pe data de 14 februarie 1924, denumirea sa a fost schimbată în ***International Business Machines Corporation***.

În această perioadă compania s-a extins atât din punct de vedere al varietății produselor oferite, cât și din punct de vedere geografic. Până în 1930 au fost construite trei fabrici pe continentul european.

În timpul crizei economice din anii '30, creșterea companiei *IBM* a continuat, chiar dacă era o perioadă de recesiune economică.

În anul 1933, la *Endicott* a fost construit unul dintre cele mai moderne laboratoare de cercetare și dezvoltare din lume. Tot aici, a fost construită o școală pentru angajați.

În anul 1934 *IBM* a devenit una dintre primele companii care au oferit asigurări pe viață angajaților săi. Doi ani mai târziu, angajații companiei au început să beneficieze de concedii plătite.

În anul 1935 *IBM* a beneficiat de un contract cu guvernul american potrivit căruia compania trebuia să păstreze o arhivă cu date referitoare la 26 de milioane de angajați din *Statele Unite*. Obiectul contractului a fost numit ***cea mai mare operațiune contabilă a tuturor timpurilor*** și rezultatele au fost atât de mulțumitoare încât *IBM* a început să beneficieze de contracte cu numeroase alte departamente guvernamentale.

Al doilea război mondial

Pe timpul desfășurării celei de-a doua conflagrații mondiale, toate bunurile companiei *IBM* au fost puse la dispoziția guvernului american.

Produsele fabricate de *IBM* au cunoscut o nouă diversificare, printre altele producându-se și arme de foc. *Thomas Watson* a reușit să vândă guvernului aceste produse cu un profit de 1%. Aceste fonduri erau folosite pentru a sprijini văduvele și copiii rămași orfani în urma deceselor angajaților *IBM* care au participat la război.

Anii războiului au însemnat primii pași ai companiei în domeniul calculatoarelor. În anul 1944, după șase ani de eforturi, a fost construit, în colaborare cu *Universitatea Harvard*, *Automatic Sequence Controlled Calculator*, cunoscut mai ales sub denumirea de *Mark I*. Aceasta a fost prima mașină care reușea să efectueze automat calcule lungi. *Mark I* cântărea peste cinci tone, avea o lungime de aproape douăzeci de metri și o înălțime de aproape trei metri. O adunare era efectuată în mai puțin de o secundă, o înmulțire dura aproximativ șase secunde, în timp ce o împărțire necesita peste zece secunde.

După terminarea războiului, în 1948, *IBM* a introdus un nou produs de acest tip și anume *Selective Sequence Electronic Calculator*, prima mașină digitală de calcul a companiei.

În același an a fost lansat produsul de succes 604 *Electronic Calculating Punch*, care s-a vândut în peste 5600 de exemplare de-a lungul celor zece ani în care a fost produs.

Primul produs destinat în exclusivitate centrelor de calcul a fost lansat în 1949 sub numele de *Card-Programmed Electronic Calculator*.

Schimbul de generații

În perioada de după cel de-al doilea război mondial au fost introduse numeroase noutăți tehnologice.

În anul 1952 a fost lansat calculatorul ***IBM 701*** care reușea să efectueze 17.000 de instrucțiuni pe secundă și era folosit doar de către cercetători și de *Guvern*.

În același an, *Thomas J. Watson* a părăsit compania după aproape 40 de ani. La conducere a urmat fiul acestuia, *Thomas J. Watson Jr*; în 1956 el a devenit *chief executive officer* al *IBM*, cu doar șase săptămâni înaintea morții tatălui său.

Dezvoltarea calculatoarelor a continuat, una dintre cele mai importante inovații fiind introducerea tranzistoarelor. În 1959 a fost produs *IBM* **7090**, care reușea să efectueze 229.000 de operații pe secundă. Produsul a fost folosit de aviația americană la construirea unui sistem de detecție a rachetelor balistice.

În anul 1957 a fost dezvoltat *RAMAC* (*Random Access Method of Accounting and Control*), primul sistem de stocare cu acces aleator.

În același an *IBM* a lansat limbajul de programare ***FORTRAN*** (*Formula Translation*), un limbaj de programare bazat pe reguli algebrice, gramaticale și sintactice care a devenit unul dintre cele mai folosite limbaje pentru calculatoare.

Era calculatoarelor

Thomas J. Watson Jr. și-a dat seama de potențialul pe care îl au calculatoarele și a orientat activitatea companiei spre acest domeniu; astfel, *IBM* a devenit liderul industriei calculatoarelor.

La data de 7 aprilie 1964, *IBM* a lansat ***System/360***, prima familie de calculatoare care foloseau echipamente perferice și permiteau modificarea *software*-ului.

O schimbare strategică importantă a avut loc în anul 1969 când *IBM* a decis că e mai bine să nu ofere doar sisteme complete, ci să vândă individual fiecare componentă *hardware* și fiecare componentă *software*. Astfel au apărut industriile de *software* și servicii.

În anul 1971 *Watson Jr.* a renunțat la funcția de *CEO* și, după un interimat de doi ani al lui *Vincent Learson*, conducerea companiei a fost preluată de ***Frank T. Cary*** în 1973.

Fostul *CEO* a devenit, în perioada 1979-1981, ambasador al *Statelor Unite* în *Uniunea Sovietică*. Până în anul 1984 și-a păstrat o funcție de conducere în consiliul director; a decedat în anul 1993.

În anul 1971 au fost introduse primele *floppy-disk*-uri care au devenit în scurt timp dispozitivele cel mai des folosite pentru păstrarea datelor.

În anul 1973, *IBM* a construit primele cititoare de coduri de bare care foloseau prisme și lentile de sticlă, precum și raze *LASER*.

Calculatorul personal

Numirea din anul 1981 a lui ***John R. Opel*** în funcția de *CEO* al companiei a coincis cu o nouă revoluție în domeniul calculatoarelor.

Datorită apariției *PC*-ului, sigla *IBM* a început să pătrundă în casele oamenilor, în școli și în companiile de dimensiuni mici.

Produsul *IBM PC* oferea toate facilitățile puse la dispoziție de un calculator, dar dimensiunile sale erau foarte reduse.

Utilizatorul avea la dispoziție o memorie de 16 KB, care putea fi extinsă până la 256 KB. El putea opta pentru una sau două unități pentru dischete și, eventual, pentru un monitor color.

Apariția PC-ului a reprezentat o altă premieră importantă pentru *IBM*. A fost pentru prima dată când componente ale unui produs care purta sigla *IBM* erau fabricate de alte companii. Procesoarele calculatoarelor personale erau produse de *Intel*, în timp ce sistemul de operare folosit, numit *DOS* (*Disk Operating System*), a fost dezvoltat de *Microsoft*, compania care avea, în acea perioadă, doar 32 de angajați.

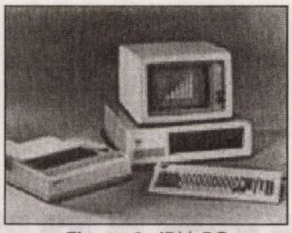


Figura 1: IBM-PC

În anul 1985 *John F. Akers* a devenit *CEO* al companiei; în perioada în care el și-a exercitat această funcție au fost alocate numeroase fonduri pentru cercetare. Ca urmare, angajații *IBM* au obținut patru premii *Nobel* pentru fizică. De asemenea, au fost obținute rezultate spectaculoase în diferite alte domenii cum ar fi matematica, stocarea datelor, telecomunicațiile etc. Astfel, au fost realizate progrese semnificative ale facilităților oferite de calculatoare.

În 1985 compania a lansat prima sa rețea locală de calculatoare. Folosind facilitățile oferite de aceasta, mai mulți utilizatori aflați în interiorul aceleiași clădiri sau al aceluiași complex puteau schimba informații sau le puteau folosi în comun. De asemenea, anumite resurse (fișiere, imprimante etc.) puteau fi folosite în comun. Astfel, *IBM* a pus bazele dezvoltării aplicațiilor pentru rețele.

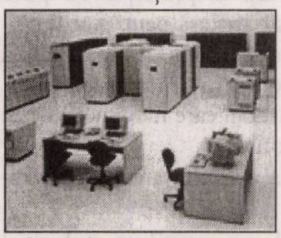


Figura 2: Rețea IBM

Ultima perioadă

Revoluția calculatoarelor personale a adus calculatoarele în casele a milioane de utilizatori. Ulterior, revoluția tehnologiei *client/server* a permis utilizatorilor să își conecteze *PC*-urile (clienții) la calculatoare performante (serverele) care funcționau în fundal și furnizau clienților aplicații și date.

Ambele revoluții au transformat modul în care utilizatorii priveau, foloseau și cumpărau produsele care foloseau ultimele tehnologii și au avut o foarte mare influență asupra dezvoltării companiei *IBM*.

Din nefericire pentru această companie, au avut loc schimbări fundamentale pe piață. Deciziile de achiziționare nu se mai luau la nivelul de vârf al companiilor, ci la nivelul departamentelor sau chiar al angajaților individuali. Astfel, compania *IBM* a pierdut majoritatea clienților tradiționali.

Comercializarea soluțiilor integrate a pierdut teren în fața comercializării componentelor. Se punea accentul pe soluții destinate indivizilor și nu pe soluții pentru companii.

Toate aceste schimbări au avut un efect negativ asupra companiei. În anul 1993 s-a înregistrat un record al pierderilor în valoare de opt miliarde de dolari.

Pe data de 1 aprilie 1993 *Louis V. Gerstner Jr.* a devenit *chairman* și *CEO* al companiei. A fost pentru prima dată când conducerea companiei a fost preluată de o persoană care nu provenea din interiorul firmei.

Imediat după sosirea sa, *Gerstner* a fost nevoit să pună în practică câteva acțiuni dramatice pentru a stabiliza compania. Au fost modificate liniile de producții, numărul angajaților a fost redus și s-au depus eforturi semnificative pentru reducerea costurilor.

Deși existau presiuni ca *IBM* să fie împărțită în mai multe companii independente, *Gerstner* a decis să păstreze integritatea acesteia. El considera că principalul avantaj al companiei era posibilitatea de a oferi clienților soluții integrate, spre deosebire de firmele concurente care furnizau în principal componente. Prin divizarea companiei s-ar fi pierdut acest avantaj unic pe care îl avea *IBM*.

După revoluția rețelelor de calculatoare și a *Internetului*, *IBM* a trebuit să suporte o nouă schimbare dramatică a pieței. De data aceasta, compania a fost mai bine pregătită pentru a face față noilor cerințe.

Clienții au devenit din nou interesați în soluții integrate, *IBM* având un avantaj important în fața concurenței din acest punct de vedere. Utilizatorii puteau beneficia de experiența *IBM* în domeniul soluțiilor oferite, al serviciilor, al produselor și al noilor tehnologii.

La expoziția *COMDEX* din toamna anului 1995, *Gerstner* a prezentat noua viziune a companiei *IBM* asupra pieței. Se punea accentul asupra soluțiilor pentru rețele de calculatoare, strategia companiei axându-se pe acest domeniu.

În mai 1997 *IBM* a demonstrat puterea extraordinară la care pot ajunge sistemele de calcul. A fost prezentat supercalculatorul *Deep Blue* care reușea să joace șah la nivelul unui campion mondial în acest sport.

Într-o partidă care a constat din șase jocuri care s-a desfășurat la *New York*, *Deep Blue* a reușit să îl învingă pe campionul mondial *Garry Kasparov* cu scorul de **3,5 - 2,5**.

A fost pentru prima dată când o mașină reușea să învingă într-o competiție de tipul turneelor de șah. Avantajul principal al calculatorului era capacitatea de a analiza 200 de milioane de poziții pe secundă.

La începutul acestui secol, *IBM* a redevenit unul dintre cei mai importanți inovatori în domeniul tehnologiei informației. Compania a avut o contribuție deosebită la realizarea revoluției *e-business*.

Concluzie

IBM a fost compania care și-a făcut simțită prezența în toate momentele cheie ale dezvoltării industriei tehnologiei informației. O mare parte a componentelor care se află în calculatoarele actuale există datorită invențiilor și inovațiilor introduse de această companie în zecile de ani de cercetări în domeniu.

Carmen Șușman este redactor GInfo și poate fi contactată prin e-mail la adresa **schummy@email.ro**.

Tehnici de CRIPTARE

Claudiu Soroiu

În acest număr continuăm prezentarea tehnicilor de criptare cu descrierea metodelor Vigenere si XOR. De asemenea, veți putea citi câteva informații referitoare la criptanaliză și la securitatea datelor.

Criptarea Vigenere

Tehnica de criptare *Vigenere* este o tehnică polialfabetică, în sensul că, prin criptare, două sau mai multe caractere diferite ale textului inițial pot fi codificate folosind același caracter în textul criptat.

Pentru a cripta un text folosind această metodă se alege mai întâi un alfabet, astfel încât toate caracterele conținute în textul inițial să se afle printre caracterele alfabetului și apoi o parolă formată din caractere ale alfabetului. Nu este necesar ca alfabetul ales să conțină numai caracterele textului inițial.

În continuare este prezentat modul în care se realizează criptarea unui text folosind tehnica *Vigenere*.

Algoritmul de criptare

Considerăm alfabetul format din literele: **A B C D E F**. Primul pas al algoritmului constă în aranjarea literelor alfabetului într-o matrice pătratică cu L linii și L coloane, unde L este numărul de simboluri ale alfabetului.

Prima linie a matricei este formată din literele alfabetului în ordine lexicografică, iar următoarele L - 1 linii sunt obținute fiecare din linia precedentă printr-o permutare la stânga cu o poziție.

În cazul alfabetului ales matricea obținută este următoarea:

	1	2	3	4	5	6
1	A	B	C	D	E	F
2	B	C	D	E	F	A
3	C	D	E	F	A	B
4	D	E	F	A	B	C
5	E	F	A	B	C	D
6	F	A	B	C	D	E

Se observă că un caracter nu apare decât o singură dată pe o linie sau o coloană și fiecare linie sau coloană conține toate literele alfabetului considerat.

Următorul pas constă în alegerea unei parole formată din simboluri ale alfabetului. Să presupunem că dorim să criptăm textul **FAAD**, folosind parola **BAB**.

Pentru ca această tehnică să fie eficientă, ar fi ideal ca lungimea parolei să fie mai mare sau cel puțin egală cu lungimea textului care urmează a fi criptat. Datorită faptului că o astfel de parolă nu poate fi reținută de către utilizator, se alege o parolă mai scurtă asupra căreia se aplică diverse metode prin care ea este extinsă pentru ca lungimea ei să depășească lungimea textului. Cea mai simplă metoda utilizată în acest sens constă în concatenarea parolei cu ea însăși până în momentul în care lungimea sa este cel puțin egală cu lungimea textului.

Pentru exemplul luat, parola, prin concatenare cu ea însăși, devine **BABBAB**.

La ultimul pas, fiecare al k-lea caracter din textul inițial este criptat astfel:

- se determină numărul de ordine i, corespunzător poziției caracterului în alfabetul considerat;
- se determină numărul de ordine j, corespunzător poziției în alfabetul considerat a celui de-al k-lea caracter din parolă;
- se scrie caracterul care se află pe linia i și coloana j în matrice.

Textul considerat, prin criptare, folosind parola extinsă **BABBAB**, devine **AABE**. Pentru $k = 1$, caracterul de pe prima poziție a textului este **F** ($i = 6$), caracterul de pe prima poziție a parolei este **B**, deci ($j = 2$).

Așadar, caracterul care va apărea pe prima poziție a textului criptat se află pe coloana a doua a celei de-a șasea linii din matrice: **A**.

Acest procedeu continuă pentru toate valorile k mai mici sau egale cu lungimea textului.

Analizând textul inițial și cel criptat se observă că primele două caractere **F** și **A**, devin prin criptare, același caracter, adică **A**, iar cele două caractere **A** de pe pozițiile 2 și 3 sunt codificate diferit în textul criptat.

Algoritmul de decriptare

Algoritmul folosit pentru decriptarea unui mesaj codificat utilizând tehnica *Vigenere* nu este foarte complex.

Primul pas care trebuie parcurs pentru a decripta un text constă în construirea parolei extinse pe baza parolei folosite. Procedeul prin care se realizează extinderea parolei în procesul de decriptare trebuie să fie același cu cel folosit pentru a extinde parola în procesul de criptare.

Pentru a decripta un text se aplică următorul algoritm: pentru fiecare al k-lea caracter din parola extinsă se găsește numărul de ordine j al acestuia în cadrul alfabetului considerat, se caută pe coloana j din matrice al k-lea caracter din textul criptat și se scrie caracterul care are numărul de ordine i, unde i este numărul liniei pe care a fost găsit caracterul.

Considerăm alfabetul definit anterior; în procesul de decriptare se ține cont de matricea atașată acestui alfabet.

Pentru textul **AABE** care a fost criptat folosind parola **BAB** (extinsă prin concatenare cu ea însăși, această parolă devine **BABBAB**) prin decriptare se obține textul **FAAD** astfel: pentru primul caracter din parola extinsă k este egal cu 1; pe coloana $j = 2$ corespunzătoare numărului de ordine al caracterului **B** din parolă se caută caracterul **A** din text și se găsește pe linia $i = 6$ și în final se scrie caracterul **F** care are numărul de ordine 6 în cadrul alfabetului. Acest procedeu se aplică pentru toate caracterele textului criptat.

Concluzii

Se observă că în cazul extinderii unei parole cu ea însăși, tehnica de criptare *Vigenere* este similară cu criptarea lui *Cezar* pe grupuri de elemente.

Dacă parola inițială are lungimea n și considerăm un număr strict pozitiv $k < n$, atunci, dacă extragem caracterele de pe pozițiile k, $n + k$, $2 \cdot n + k$, ... etc. din textul inițial, se observă că pentru criptarea lor s-a folosit același deplasament.

Criptarea XOR

Tehnica de criptare *XOR* se bazează pe operatorul binar *XOR* (*eXclusive OR - sau exclusiv*). Reamintim faptul că rezultatul aplicării acestui operator asupra a două valori binare identice este 0, în timp ce rezultatul aplicprii sale asupra a două valori binare diferite este 1. Rezultatele aplicării acestui operator sunt sintetizate în figura următoare.

⊕	0	1
0	0	1
1	1	0

Algoritmul de criptare

Pentru a cripta un mesaj folosind această metodă este nevoie de o parolă și de un alfabet. În acest caz, numărul de simboluri ale alfabetului trebuie să aibă forma 2^n, unde n este un număr natural.

Și de această dată, ar fi ideal ca lungimea parolei să fie mai mare sau egală cu lungimea textului care urmează a fi criptat. Ca urmare a acestui fapt, primul pas al algoritmului de criptare constă în obținerea parolei extinse, la fel ca în cazul criptării *Vigenere*.

După obținerea parolei extinse, procedeul de criptare este: pentru fiecare al k-lea caracter din text și al k-lea caracter din parolă se caută în cadrul alfabetului numerele de ordine i și j ale celor două caractere, se aplică operatorul *XOR* asupra reprezentării binare a celor două numere (se obține reprezentarea binară a unui număr n) și, în final, se scrie caracterul care are numărul de ordine n în cadrul alfabetului.

Fie alfabetul format din simbolurile setului de caractere *ASCII* extins, care are 2^8 elemente. Dacă dorim să criptăm un text, și presupunem că pe poziția k avem caracterul **A** care are numărul de ordine 65, iar în parola extinsă avem pe poziția k caracterul **B** cu numărul de ordine 66, atunci, prin aplicarea operatorului *XOR* între reprezentările binare ale celor două numere, se obține caracterul care are numărul de ordine 3. În cadrul setului de caractere *ASCII* extins, caracterele cu numere de ordine mai mici decât 32 poartă numele de *caractere de control*, și sunt considerate caractere neimprimabile. De obicei, dacă se folosește standardul *ASCII*, după criptarea cu această metodă, foarte puține caractere sunt imprimabile.

Aplicarea operatorului *XOR* asupra reprezentării binare a numerelor 65 și 66 este ilustrată în cele ce urmează:

```
0 1 0 0 0 0 0 1 = 65
0 1 0 0 0 0 1 0 = 66
0 0 0 0 0 0 1 1 =  3
```

Algoritmul de decriptare

Algorimul de decriptare este asemănător cu cel de criptare, singura deosebire constă în faptul că mesajul criptat ia locul textului inițial. Să presupunem că în textul criptat avem pe poziția k caracterul cu numărul de ordine 3 și în parola extinsă avem pe poziția k caracterul cu numărul de ordine 66 (**B**), prin aplicarea operatorului *XOR* asupra reprezentării binare ale celor două numere se obține caracterul cu numărul de ordine 65 (**A**), astfel:

```
0 0 0 0 0 0 1 1 =  3
0 1 0 0 0 0 1 0 = 66
0 1 0 0 0 0 0 1 = 65
```

În cazul în care nu se cunoaște parola cu care a fost criptat un mesaj, dar se cunoaște faptul că acesta conține foarte multe caractere imprimabile, se presupune că parola are cel puțin $n \geq 3$ caractere și se încearcă aplicarea operatorului *XOR* între textul criptat și un caractere imprimabil astfel încât, pentru toate numerele întregi strict pozitive $k < n$, în mulțimea formată cu caracterele de pe pozițiile k, $n + k$, $2 \cdot n + k$, ..., etc., să se obțină un maxim de caractere imprimabile care pot să facă parte din textul inițial. În cazul în care nu s-au obținut rezultatele dorite numărul n se incrementează cu 1 și se reia pasul anterior.

După cum se poate observa, și această tehnică de criptare este polialfabetică.

Dacă asupra unui text se aplică succesiv mai multe tehnici de criptare diferite, atunci procesul de decriptare este îngreunat foarte mult datorită faptului că dacă un criptanalist încearcă să decripteze textul pe baza frecvenței de apariție a literelor, el are foarte puține șanse de reușită, mai

ales dacă una dintre tehnicile de criptare utilizate operează cu două alfabeturi distincte și de lungimi diferite.

Criptanaliză

Criptanaliza este știința care se ocupă cu studiul decriptării mesajelor când nu se cunoaște cheia de criptare.

Există mai multe procedee de analiză a textelor criptate. Cele mai des utilizate sunt analiza frecvenței de apariție a caracterelor și analiza probabilistică. De exemplu, în limba engleză, caracterul cel mai frecvent folosit este **E** iar grupurile de două caractere care apar cel mai des sunt **TH** și **HE**.

Metodele matematice care stau la baza criptanalizei și criptografiei moderne sunt foarte complexe.

Pentru a decripta un text este nevoie de cel puțin două informații. Prima informație constă în cunoașterea metodei generale de criptare utilizată sau categoria de tehnici de criptare din care face parte, iar a doua informație constă din cunoașterea unor parametri implicați în procesul de criptare (informații asupra modului de utilizare al cheii de criptare).

În practică, utilizatorii criptografiei dețin în cele mai multe cazuri, echipamente pentru criptarea și decriptarea datelor pentru o anumită tehnică sau categorie de tehnici de criptare.

Securitatea datelor în zilele noastre

Dacă dorim să accesăm un cont de pe un *server* avem nevoie de o parolă. Confidențialitatea mesajelor și a datelor transmise de la *client* la *server* și invers depinde de gradul de complexitate al metodei de criptare folosite de cele două sisteme *client*, respectiv *server* și de complexitatea parolei alese.

Chiar dacă se folosește o parolă complexă, utilizarea unui cod unic de acces prezintă unele dezavantaje. În cazul în care parola este interceptată de anumite persoane, complexitatea sa nu mai are o importanță foarte mare. Cea mai sigură modalitate de acces la un *server* ar fi ca la fiecare accesare a contului să se utilizeze o altă parolă. Acest tip de metode poartă numele de ***one time pass*** (*parolă folosită o singură dată*).

În cele ce urmează vom prezenta un exemplu de realizare a unui sistem ce se încadrează în tipul *one time pass*.

Prezentare

În cazul parolelor care se folosesc o singură dată, trebuie utilizată o tehnică de criptare a datelor. De obicei, parolele sunt constituite din caractere imprimabile, deci alfabetul utilizat trebuie să fie o submulțime a setului de caractere imprimabile din întregul set de caractere utilizat de calculator.

În general, dacă dorim să criptăm un text T, ale cărui litere aparțin unui alfabet A, avem nevoie de:

- o funcție de criptare f definită pe mulțimea părților textului T;
- o parolă sau cheie de criptare.

De obicei, caracterele din textul inițial și cel criptat aparțin aceluiași alfabet.

Dintr-un alfabet pot lipsi semnele obișnuite de punctuație și spațiile, cuvintele fiind scrise unul după altul, lucru care îngreunează foarte mult munca celui care încearcă să decripteze mesajul criptat în eventualitatea în care nu deține parola folosită în procesul de codificare a mesajului.

Textul inițial T și textul criptat C sunt împărțite în mai multe grupuri compuse din una, două sau mai multe litere, în funcție de tehnica de criptare utilizată. În tehnicile prezentate anterior cele două texte erau împărțite în grupuri de câte o singură literă, deoarece transformările erau aplicate la un moment dat, asupra unui singur caracter din text.

Funcția de criptare f este definită pe mulțimea tuturor grupurilor de litere care se formează din textul inițial, și, de obicei, este o funcție bijectivă. Inversa ei, f^{-1}, se folosește pentru a decripta mesajul.

Fie f_P funcția de criptare asociată parolei P. Pentru o parolă indrodusă către utilizator procesul de criptare este dat de formula $f_P(T) = C$.

Pentru a decripta textul C, de obicei, se folosește funcția inversă f_P^{-1} asociată parolei P, și în procesul de decriptare se folosește formula $f_P^{-1}(C) = T$.

Un *sistem one time pass* este un sistem în care conexiunea dintre *client* și *server* se face prin folosirea unui model *one time pass*. În cazul unui sistem *one time pass* se alege o parolă inițială P_0 care este stocată pe *server* și un număr întreg strict pozitiv n suficient de mare. Numărul n poate să aibă o valoare implicită, fără a mai necesita alegerea lui de către utilizator. Funcția de criptare, pe care atât clientul cât și *server*-ul o aplică parolei P_0, trebuie să fie bijectivă, iar inversa ei trebuie să fie foarte greu de calculat folosind procedee matematice.

Clientul deține, la rândul său, parola inițială P_0 și numărul n. Când se realizează conexiunea dintre *client* și *server*, clientul aplică funcția de criptare f de n ori astfel:

$$P_i = f(P_{i-1}),\ i = 1, ..., n.$$

Parola P_n care se obține este transmisă *server*-ului. La rândul său *server*-ul creează, utilizând același procedeu, propria sa parolă P_n, și dacă coincide cu cea pe care a trimis-o clientul se realizează conexiunea și numărul n este decrementat automat cu 1 atât pe stația *server* cât și pe stația *client*. În acest mod, la următoarea accesare a contului de pe *server*, se va folosi o altă parolă.

Modelul *one time pass* presupune faptul că toate cele n parole care se obțin prin aplicarea succesivă a funcției f, sunt distincte. În momentul în care s-au epuizat toate cele n parole distincte, trebuie înlocuită cheia (parola) P_0.

Dacă funcția f ar fi ușor inversabilă, atunci, dacă parola curentă este interceptată de o persoană neautorizată, prin aplicarea funcției f^{-1} se obține parola care va fi utilizată la următoarea accesare a contului de pe *server*.

Funcția f trebuie aleasă astfel încât, chiar dacă expresia sa intră în posesia unor persoane neautorizate, expresia funcției f^{-1} să fie foarte greu de calculat.

Claudiu Soroiu este redactor al GInfo. Poate fi contactat prin e-mail la adresa **csoroiu@yahoo.com**.

CABLAREA edificiilor

Mihai Scorțaru

În cadrul acestui articol vom prezenta principalele caracteristici fizice ale mediilor de comunicații folosite pentru realizarea rețelelor de calculatoare. De asemenea, vom prezenta standardul EIA 568A care descrie modul în care trebuie realizată cablarea elementelor de telecomunicații din clădirile de dimensiuni medii.

Medii de transmisie

Pentru ca informațiile să poată fi transmise de la un calculator la altul, trebuie să existe un mediu fizic prin intermediul căruia acestea să circule. Aceste medii fizice poartă denumirea de *medii de transmisie*.

Acestea trebuie să asigure performanțe cât mai ridicate din punct de vedere al vitezei de transmisie, ratei de erori, costului etc.

Mediile de transmisie pot fi grupate în două mari categorii:

- *hardwire* (bazate pe fir);
- *wireless* (nebazate pe fir).

Medii hardwire

Mediile de transmisie bazate pe fir pot fi grupate, la rândul lor, în două categorii:

- *electrice* (folosesc cablul electric);
- *optice* (folosesc fibra optică).

Pentru a oferi performanțe ridicate la transmiterea datelor, cablurile electrice folosite la cablarea rețelelor trebuie să aibă rezistență, capacitate și impedanță reduse, flexibilitate ridicată, precum și rezistență la tracțiune și temperatură.

Caracteristicile transmisiei folosind aceste cabluri depind de geometria secțiunii, numărul de conductori, distanța dintre aceștia, tipul izolatorului folosit, materialul folosit pentru protecția electrică etc.

În cele ce urmează vom prezenta principalii parametri care caracterizează cablurile electrice:

- *Impedanța* cablului este cel mai important parametru electric; valoarea acesteia este, de obicei, cuprinsă între 50 și 150 Ω. Cea mai folosită impedanță este cea de 100 Ω.
- *Viteza de propagare* a semnalului este parametrul care stă la baza determinării vitezei de transmisie a datelor. De obicei, viteza este cuprinsă între 0,55 · *c* și 0,75 · *c*, unde *c* este viteza luminii (300.000 km/s). Putem considera o viteză medie egală cu 0,66 · *c*, adică aproximativ 200.000 km/s.
- *Atenuarea* (pierderea) suferită de semnal pe cablu este un parametru important, mai ales dacă se lucrează la frecvențe mari. În acest cazuri, valoarea atenuării (exprimată în *dB*) trebuie să fie cât mai mică.
- *Diafonia* este un parametru care descrie influența unui cablu asupra unui alt cablu aflat în apropiere. Diafonia reprezintă raportul (exprimat în *dB*) dintre valoarea unui semnal prezent pe un cablu și valoarea semnalului indus pe un cablu vecin. Pentru a limita acest efect se folosește *ecranarea* cablurilor prin folosirea unui înveliș metalic legat la pământ. Ecranarea poate fi realizată pentru fiecare conductor care formează cablul, sau pentru întreg cablul. În unele cazuri se folosesc ambele modalități.
- *Geometria secțiunii* este un parametru care descrie diametrul cablului. Diametrul poate fi exprimat în milimetri, dar pe plan internațional se folosește unitatea de măsură *AWG* (*American Wire Gage*). *AWG* este o scală cu regresie geometrică, valorile posibile fiind cuprinse între 0 și 38 *gage*. Cablurile folosite în trasmiterea datelor au secțiuni cuprinse între 22 și 26 *AWG*.
- *Siguranța în caz de incendiu* este o normă care trebuie respectată, normă stabilită la nivelul fiecărui stat. De obicei se folosesc cabluri cu proprietăți speciale cum ar fi:
  - întârzierea propagării flăcării;
  - reducerea fumului produs în caz de incendiu;
  - reducerea cantității de gaze toxice emise.

Există două categorii importante de cabluri electrice folosite pentru transmiterea datelor: *cablul torsadat* și *cablul coaxial*.

Cablul torsadat este mediul tradițional folosit în telefonie, este bazat pe cupru și mai este numit *cablu cu perechi de fire răsucite* (*Twisted Pair - TP*).

Perechea de fire răsucite este formată din două fire metalice din cupru, răsucite după un anumit pas. Diametrul uzual este de 24 *AWG* (0,5 mm), iar pasul de răsucire depinde de numărul de perechi care formează cablul. O valoare des folosită pentru pasul de răsucire este de 1/6 *inci*.

Răsucirea firelor reduce distorsiunile cauzate de câmpurile electromagnetice externe, deoarece acestea vor acționa în mod egal asupra celor două fire.

Dacă se folosește un cablu format din mai multe perechi de fire răsucite, atunci pașii de răsucire trebuie să difere pentru fiecare pereche în parte. Valorile acestora trebuie să fie alese în așa fel încât diafonia să fie redusă cât mai mult posibil.

Din punct de vedere al metodei de ecranare folosite, există trei categorii de cabluri *TP*:

- *STP* (*Shielded Twisted Pair*) - ecranare pentru fiecare pereche de fire și ecranare globală pentru întreg cablul;
- *FTP* (*Foiled Twisted Pair*) - doar ecranare globală pentru întreg cablul;
- *UTP* (*Unshielded Twisted Pair*) - fără ecranare.

Din punct de vedere al tipurilor de transmisii în care se folosesc, cablurile torsadate pot fi clasificate în cinci categorii:

- *Categoria I* (*Telecommunication*) - cabluri folosite în telefonia analogică clasică;
- *Categoria a II-a* (*Low Speed Data*) - cabluri folosite în telefonia analogică și digitală care oferă transmisii de date la viteze reduse;
- *Categoria a III-a* (*High Speed Data*) - cabluri folosite în realizarea rețelelor locale cu rate de transfer de până la 10 *Mbps*;
- *Categoria a IV-a* (*Low Loss, High Performance Data*) - cabluri cu performanțe ridicate din punct de vedere al atenuării și al vitezei de transmiei; sunt folosite în rețele cu rate de transfer de ordinul zecilor de *Mbps*.
- *Categoria a V-a* (*Low Loss, Extended Frequency, High Performance Data*) - cabluri folosite în relizarea rețelelor locale cu rate de transfer de până la 100 *Mbps*.

Cablurile de categoria a VI-a și a VII-a se află în atenția cercetătorilor. Acestea vor oferi rate de transfer de 200 *Mbps*, respectiv 600 *Mbps*.

Cablul coaxial este folosit, în principal, ca mediu de transmisie pentru rețelele locale de tip *Ethernet*. Performanțele acestuia au fost atinse de cablul torsadat pentru distanțe mici și au fost depășite de cablul cu fibră optică pentru distanțe mari. Cablul coaxial este format dintr-un miez de cupru numit ***conductor central***, un strat izolator, un ***conductor exterior*** și o teacă de plastic.

Pentru realizarea transmisiilor de date este folosit ***cablul coaxial în bandă largă*** care are o impedanță de 50 Ω.

Există două categorii de astfel de cabluri:

- ***cablu Ethernet gros*** - deși parametri electrici sunt foarte buni, datorită rigidității și a razei de curbură maxim admise, el este dificil de montat, cablajul interiorului unei clădiri fiind costisitor;
- ***cablu Ethernet subțire*** - este foarte răspândit deoarece oferă un raport preț/performanță foarte bun.

Conectarea calculatoarelor la cablul coaxial poate fi realizată prin două metode:

- prin folosirea joncțiunilor *T* - implică tăierea cablului;
- prin folosirea conectorilor de tip ***vampir*** - permite conectarea fără a fi necesară tăierea cablului.

De obicei, metoda folosită pentru transmiterea de date utilizând cablul coaxial este ***Manchester*** sau ***Manchester diferențial***.

Cablul cu ***fibră optică*** este format dintr-un fir de material sticlos numit ***nucleu*** sau ***miez***, acoperit cu un înveliș de sticlă numit ***manta***. Proprietățile de difracție ale celor două medii de sticlă trebuie să fie diferite. La exterior ele sunt protejate de un înveliș de plastic.

Diametrul nucleului este de aproximativ 125 μm, iar diametrul total al firului de fibră optică este de 250 μm.

Proprietățile datorită cărora fibra optică este mediul ideal pentru rețelele de mare viteză și întindere medie sunt următoarele:

- atenuarea redusă - permite lungimi de ordinul kilometrilor ale tronsoanelor de fibră optică;
- imunitate totală la efectele câmpurilor electromagnetice - elementele fizice de transport nu au sarcină electrică; informațiile sunt transmise cu ajutorul *fotonilor*, particule neutre din punct de vedere electric;
- capacitate mare de transmisie - se pot atinge viteze de transfer de ordinul gigabiților pe secundă;
- dimensiune redusă, greutate mică, flexibilitate mare - nu ridică probleme la cablarea edificiilor.

Datorită faptului că mediul de transmisie este optic, ar putea apărea pierderi de informații datorită refracției semnalului luminos la suprafața de separație dintre nucleu și manta. Soluția este dată de ***legea lui Snell*** potrivit căreia, pentru valori ale unghiului de incidență mai mari decât valoarea unghiului critic, reflexia este totală, iar refracția este nulă.

Valoarea unghiului critic este dată de formula:

$$\theta = \arcsin(n_1 / n_2),$$

unde n_1 și n_2 sunt indicii de refracție a celor două medii.

De obicei, indicele de refracție al nucleului este $n_1 = 1{,}5$, iar indicele de refracție al mantalei este $n_2 = 1{,}457$. Se obține un unghi critic cu o valoare de aproximativ 79,5°.

Energia luminoasă se poate propaga în interiorul fibrei optice într-un număr finit de configurații. Fiecare astfel de configurație, numită ***mod***, are propriile proprietăți de propagare. Fibra optică ce permite folosirea mai multor moduri de propagare poartă denumirea de fibră ***multimodală***. Razele de lumină care se propagă după moduri diferite vor avea timpi de propagare diferiți. Acest fenomen poartă denumirea de ***dispersie modală***.

Dispersia modală poate fi redusă dacă se folosește o fibră monomodală al cărei nucleu are un diametru aproximativ egal cu cel al lungimii de undă al razei luminoase (valori cuprinse între 8 și 10 μm).

Fibrele optice pot fi folosite doar pentru realizarea de conexiuni *punct la punct*. Pentru realizarea acestor conexiuni sunt necesare dispozitive opto-electronice. Pentru generarea fluxului luminos sunt folosite *LED*-uri (*Light Emitting Diode* - diode emițătoare de lumină), iar pentru receptarea acestuia sunt folosite ***fotodiode***.

Atenuarea semnalului luminos depinde de lungimea de undă a acestuia. Pentru comunicațiile optice se folosesc lungimi de undă din afara spectrului vizibil, și anume în gama 750-1600 nm.

În cadrul acestui interval s-au definit trei *ferestre* de lungimi de undă:

- *prima fereastră* este definită în jurul lungimii de undă de 850 nm; este folosită de fibrele multimodale cu *LED*-uri, frecvența semnalului optic poate atinge 150 MHz, iar atenuarea este de aproximativ 3,5 dB/km;
- *a doua fereastră* este definită în jurul lungimii de undă de 1300 nm; este folosită de ambele tipuri de fibre, transmisia poate fi *LASER* (frecvențe de până la 1 GHz) sau prin *LED*-uri (frecvențe de până la 500 MHz), iar atenuarea este de sub 1 dB/km;
- *a treia fereastră* este definită în jurul lungimii de undă de 1550 nm; este folosită de fibrele unimodale cu *LASER*, frecvența semnalului optic poate atinge 100 GHz, iar atenuarea este cuprinsă între 0,2 și 0,4 dB/km.

Datorită atenuării reduse, fibra optică poate fi folosită pentru transmisii pe distanțe mari. Atenuarea admisă este de 10-20 dB, așadar lungimea unei fibre optice multimodale poate atinge 10 km fără a fi necesară amplificarea semnalului. Pentru fibrele unimodale, distanța maximă poate atinge 100 km.

Cablul cu fibră optică este de mai multe tipuri, la cablarea rețelelor locale fiind folosite următoarele categorii:

- *cablul de tip "dens" (tight)* - folosit pentru cablarea interioară a clădirilor; nucleele sunt protejate de mantale rezistente la foc și cu degajare redusă de gaze toxice; în centrul cablului se află un material dielectric;
- *cablul de tip "rarefiat" (loose)* - folosit pentru cablări în exteriorul clădirilor; trebuie protejat împotriva umidității;
- *cablul slotted core* - folosit tot pentru cablări în exteriorul clădirilor, dar rezistent la umiditate.

Pentru realizarea conectării fibrei optice la calculator se folosesc două tipuri de ***interfețe*** (ansambluri foto-electrice folosite pentru conectarea la fibră a unui dispozitiv de transmitere a datelor):

- ***interfața pasivă*** este realizată cu ajutorul a două conectoare *baionetă*, înfipte în conductor; unul conține un *LED* pentru transmiterea datelor, iar celălalt conține o fotodiodă pentru recepționarea acestora;
- ***interfața activă*** sau ***repetorul activ*** cuprinde trei așa-numite etaje:
  - ♦ etajul care recepționează semnalul optic și îl transformă în semnal electric;
  - ♦ etajul care amplifică semnalul electric; la acest nivel se conectează calculatorul printr-un cablu electric;
  - ♦ etajul care transformă semnalul electric amplificat în semnal optic și îl transmite pe fibră.

În cazul interfețelor active, căderea unei componente nu va afecta transmiterea datelor pe tronson, ci doar va întrerupe conectarea acesteia la tronson.

Datorită faptului că repetoarele active acționează direct asupra semnalului și îl prelucrează, căderea sa va afecta întregul tronson.

Medii wireless

Mediile de transmisie nebazate pe fir pot fi și ele folosite pentru realizarea rețelelor locale, datorită progreselor înregistrate în ultimii ani în acest domeniu.

Principale avantaje ale folosirii mediilor *wireless* sunt date de flexibilitatea oferită la instalarea, reconfigurarea și întreținerea stațiilor.

Dezavantajele principale sunt date de ratele de transfer reduse (cel mult 16 Mbps) și de ratele ridicate ale erorilor. Erorile pot apărea datorită interferențelor sau atenuărilor cauzate de relief și condiții atmosferice.

Principalele medii *wireless* pentru transmiterea datelor sunt undele radio, microundele și razele infraroșii.

Legăturile ***radio*** acoperă o bandă de frecvențe foarte largă. În telecomunicații sunt folosite mai ales benzile de frecvențe înalte (*HF - High Frequency*) cuprinse între 3 și 30 MHz, foarte înalte (*VHF - Very High Frequency*) cuprinse între 30 și 300 MHz și ultra înalte (*UHF - Ultra High Frequency*) cuprinse între 0,3 și 3 GHz.

Utilizarea undelor *VHF* și *UHF* oferă avantajele unei propagări fără erori a semnalului radio, datorită reflexiei și refracției reduse din stratele ionosferei. Distanța maximă între stații este de ordinul zecilor de kilometri, o valoare uzuală fiind 20 km.

Principalele dezavantaje constau în posibilitățile de apariție a interferențelor și de expunere a utilizatorilor la radiații electromagnetice. De asemenea, vitezele de transfer sunt relativ mici.

Microundele acoperă gama de frecvențe cuprinse între 2 și 40 GHz. Domeniul este împărțit în benzi care sunt atribuite diferitelor companii și organizații. Din aceste motive, rețelele locale bazate pe microunde funcționează în cadrul unei benzi de frecvențe dedicate.

Viteza de transfer este comparabilă cu cea de la rețelele locale clasice, rata de erori este acceptabilă, iar interferențele lipsesc datorită faptului că banda este dedicată.

Totuși, există și unele dezavantaje: semnalul poate fi atenuat datorită condițiilor atmosferice, iar utilizatorii sunt expuși la radiații.

Razele ***infraroșii*** acoperă partea superioară a spectrului electromagnetic; frecvențele semnalelor folosite în cadrul rețelelor de acest tip sunt de ordinul sutelor de THz.

Viteza de transmisie și rata de erori sunt acceptabile, interferențele lipsesc, iar costul este redus pentru anumite aplicații.

Principalul dezavantaj îl constituie faptul că stațiile de la capetele unei legături trebuie să fie aliniate.

Elemente de cablare structurată

Standardele care descriu modul în care trebuie să se realizeze cablarea unui edificiu sau a unui grup de edificii prezintă, printre altele, numeroase detalii referitoare la:

- caracteristicile aparatelor electrice, electronice și optice;
- caracteristicile mediilor de comunicație folosite și ale elementelor pasive necesare (conectori, terminatori, adaptoare etc.);
- viteza de transmisie admisă;
- topologia cablajului (stea, inel, magistrală);
- distanțele maxime admise;
- regulile de instalare.

Aceste standarde au fost elaborate după 1985 și se bazează pe experiența unor firme consacrate în acest domeniu cum ar fi *IBM* sau *Digital*.

Principalele standarde existente sunt următoarele:

- EIA/TIA 586A - pentru cablarea edificiilor comerciale;
- EIA/TIA 570 - pentru cablarea edificiilor rezidențiale (clădiri de dimensiuni reduse);
- ISO/IEC 11801 - standard aflat în dezvoltare.

În cele ce urmează ne vom concentra atenția asupra standardului clasic (EIA/TIA 586A). Acesta a fost definitivat în 1991 și poartă denumirea de *Commercial Building Telecommunication Cabling Standard*.

Acest standard descrie modul de cablare pentru un edificiu sau un grup de edificii de dimensiuni medii. Specificațiile sale se referă la:

- condițiile care trebuie îndeplinite pentru ca acest standard să poată fi folosit în cadrul cablării unei clădiri;
- topologia cablării;
- elementele componente ale cablajului;
- mediile de transmisie folosite;
- cablarea pe verticală;
- cablarea pe orizontală;
- norme de instalare;
- identificarea cablurilor.

Sistemul de cablare structurată definit de acest standard este compus din șase subsisteme:

- *entrance facilities* - sistemul de la intrarea în clădire;
- *equipment room* - camera cu echipamente;
- *backbone* - cablarea pe verticală;
- *telecommunication closet* - dulapul pentru telecomunicație;
- *horizontal cabling* - cablarea pe orizontală;
- *work area* - componentele zonei de lucru.

Condiții necesare

Pentru ca standardul EIA/TIA 568A să poată fi utilizat, trebuie să fie satisfăcute următoarele condiții:

- distanța dintre două stații trebuie să fie de cel mult 3 km;
- suprafața acoperită de grupul de edificii trebuie să fie de cel mult 1 km^2;
- numărul de persoane din edificii trebuie să fie de cel mult 50.000;
- proiectul de cablare realizat conform standardului trebuie să fie valabil cel puțin 10 ani.

Topologia cablării

Cablarea este realizată folosind o topologie de tip stea, organizată ierarhic. Aceasta a fost aleasă datorită flexibilității sale, toate celelalte topologii (inel, magistrală etc.) pot fi simulate folosind topologia de tip stea.

Centrul topologiei este reprezentat de comutatorul principal care este unic pentru întreaga zonă cablată.

Al doilea nivel ierarhic este reprezentat de comutatoarele intermediare care corespund câte unui edificiu. Dacă se cablează un singur edificiu, atunci nu există comutatoare intermediare.

Cel de-al treilea nivel este reprezentat de oficiul pentru telecomunicații care este asociat unui etaj sau unui grup de încăperi.

Elementele cablajului

Așa cum am afirmat anterior, elementul central al topologiei este *comutatorul principal* (*main crossconnect*). Acesta este un centru de distribuție de la care pornesc cablurile principale către edificii sau, în cazul în care se cablează un singur edificiu, către diferite planuri de cablare.

Comutatoarele intermediare (*intermediate crossconnect*) se află în fiecare edificiu și reprezintă dulapurile de distribuție ale cablurilor către diferite planuri de cablare.

Oficiul pentru telecomunicații (*telecommunication closet*) reprezintă dulapul local de distribuție pentru cablurile la care se conectează stațiile sau pentru cablurile aferente cablării pe verticală.

Tronsonul interedificii (*interbuilding backbone*) identifică principalele cabluri prin intermediul cărora edificiile sunt conectate la comutatorul principal.

Tronsonul intern (*intrabuilding backbone*) identifică principalele cabluri prin intermediul cărora oficiile pentru telecomunicații sunt conectate la comutatorul intermediar al edificiului din care fac parte.

Încăperea cu echipamente (*equipment room*) corespunde unui plan de cablare și conține diferite aparate necesare pentru funcționarea subrețelei aferente planului de cablare respectiv .

Infrastructura pentru intrare (*interbuilding entrance facility*) realizează interfața dintre tronsonul interedificii și tronsonul intern. Conține diverse elemente care asigură protecția electrică (legarea la pământ, siguranțe etc.).

Zona de lucru (*work area*) identifică stațiile de lucru, cablurile prin intermediul cărora acestea sunt conectate la sistemul de cablare, diferite adaptoare etc. Practic, zona de lucru reprezintă toate elementele de comunicare din exteriorul dulapului de distribuție local. Toate aceste elemente trebuie să fie ușor accesibile, iar interconectarea lor trebuie să fie cât mai ușor de realizat.

Panourile intermediare (*patch panels*) sunt folosite pentru interconectarea mediilor de transmisie.

Blocurile de terminatori (*wiring blocks*) reprezintă terminatorii mecanici ai cablurilor.

Prizele pentru comunicație (*telecommunication outlets*) sunt prizele prin intermediul cărora stația de lucru este.

conectată la sistemul de cablaj. Fiecare astfel de priză conține doi sau mai mulți conectori (zonele de lucru trebuie să aibă cel puțin doi conectori: unul pentru voce și unul pentru date).

Mediile de transmisie

Standardul EIA/TIA 568A permite folosirea următoarelor medii de transmisie:

- cablu coaxial cu rezistența de 50Ω;
- fibră optică multimodală cu diametrul nucleului de 62,5 μm și diametrul mantalei de 125 μm;
- fibră optică multimodală cu diametrul nucleului de 8,3 μm și diametrul mantalei de 125 μm;
- cabluri torsadate *UTP* cu impedanța de 100Ω; pot avea o pereche, patru perechi sau 25 de perechi de fire; nu pot fi folosite cabluri de categoria I sau a II-a;
- cabluri torsadate *STP* cu impedanța de 150Ω.

Cablarea pe verticală

Tronsoanele pot realiza conectarea unor edificii sau a unor planuri de cablare. Standardul specifică lungimile maxime ale unui tronson în funcție de mediul de transmisie folosit. Valorile sunt prezentate în tabelul 1.

Tip	TC-MC	TC-IC	IC-MC
fibră optică	2000 m	500 m	1500 m
cablu UTP	800 m	500 m	300 m
cablu STP	700 m	500 m	700 m
cablu coaxial	500 m	500 m	500 m

Legendă

Tip: *tipul cablului;*
TC-MC: *distanța maximă dintre oficiul pentru telecomunicații și comutatorul central dacă nu există comutatoare intermediare;*
TC-IC: *distanța maximă dintre oficiul pentru telecomunicații și un comutator intermediar;*
IC-MC: *distanța maximă dintre un comutator intermediar și comutatorul central.*

Tabelul 1: Distanțe admise la cablarea pe verticală

Cablarea pe orizontală

Prin această noțiune se înțelege conectarea stațiilor de lucru la dulapul local de distribuție. Topologia acestei cablări este tot una stelară, centrul stelei fiind oficiul pentru telecomunicații, iar razele acesteia fiind legăturile către stațiile de lucru.

Distanța maximă dintre oficiul de telecomunicații și prizele pentru comunicații este de 90 m, iar distanța dintre stația de lucru și o priză trebuie să fie de cel mult 3 m.

Pentru realizarea cablării pe orizontală pot fi folosite cabluri *UTP* cu patru perechi de fire, cabluri *STP* cu două perechi, cabluri coaxiale subțire cu rezistența de 50Ω sau cabluri cu fibră optică multimodală.

Norme de instalare

Standardul EIA/TIA 568A specifică anumite norme care trebuie respectate în momentul realizării cablării. Cele mai importante restricții care trebuie respectate sunt următoarele:

- în cazul mediilor de transmisie electrice, tensiunea prezentă pe fir trebuie să aibă o valoare inferioară unui maxim specificat în funcție de tipul cablului folosit pentru realizarea conexiunilor;
- cablarea pe orizontală trebuie să permită un acces facil la fire;
- distanța la care se află cablurile folosite pentru realizarea rețelei față de cablurile de alimentare sau de putere trebuie să fie mai mare decât o valoare minimă specificată în funcție de tipul mediului de transmisie;
- izolarea mecanică a cablurilor trebuie să fie corespunzătoare pentru a evita deteriorarea acestora;
- cablurile electrice folosite trebuie protejate prin legare la pământ;
- cablurile cu fibre optice trebuie protejate prin metode specifice.

Identificarea cablurilor

Pentru a ușura întreținerea și a permite dezvoltările ulterioare, trebuie stabilită o metodologie unitară pentru identificarea cablurilor.

Pentru un cablu care face parte dintr-un tronson, identificatorul său este format din două câmpuri numerice. Primul reprezintă cablul propriu-zis, iar al doilea indică numărul de perechi de fire răsucite sau de fibre optice din care este alcătuit.

Pentru un cablu care realizează conectarea unei stații, standardul prevede un identificator alfanumeric fomat din patru câmpuri. Primul câmp indică edificiul în care se află stația de lucru, al doilea identifică oficiul de distribuție la care este conectată stația de lucru, al treilea câmp este un identificator numeric al stației, iar al patrulea indică dulapul local de distribuție.

Exemplu

Stația de lucru folosită pentru redactarea textului acestui articol ar putea avea următorul identificator: `AG0417B`.

Primul câmp este `AG` și identifică edificiul în care se află stația de lucru, adică sediul *Agora Media*.

Prin intermediul celui de-al doilea câmp (`04`) se specifică etajul la care se află stația de lucru; în acest caz stația se află la al patrulea etaj al clădirii.

Câmpul `17` este folosit pentru a identifica stația, fiind un număr de ordine.

Ultimul câmp specifică faptul că identificatorul dulapului local de distribuție este `B`.

În numărul următor...

Prezentarea rețelelor de calculatoare va continua cu descrierea principalelor standarde care specifică modul în care trebuie realizate rețelele locale clasice.

Mihai Scorțaru este redactor-șef adjunct al GInfo. Poate fi contactat prin e-mail la adresa **skortzy@yahoo.com**.

În numărul URMĂTOR

Numărul 12/4 al Gazetei de Informatică va apărea în a doua jumătate a lunii aprilie. Vă prezentăm pe scurt câteva dintre articolele pe care le veți putea citi...

Heroes of Might and Magic IV

La pagina 7 a acestui număr puteți citi o scurtă prezentare a acestui joc; în numărul următor va apărea, în premieră pentru revista noastră, primul review complet al unui joc. Datorită faptului că seria preferată de jocuri a majorității membrilor redacției este Heroes of Might and Magic, review-ul nu poate prezenta decât cel mai nou joc din această familie. Veți putea afla totul despre cele șase tipuri de orașe, despre cele 66 de creaturi, despre vrăjile care pot fi folosite, despre noile concepte de joc etc.

Microsoft

În ultimele trei numere am prezentat istoria unor companii care au influențat, mai ales, partea hardware a domeniului IT. Este timpul să prezentăm și o companie al cărei domeniu de activitate să fie producerea de software. Nu putem să începem decât cu gigantul Microsoft; așadar, în numărul următor va fi prezentat modul în care a evoluat cea mai cunoscută companie producătoare de software din lume.

Microsoft

Tehnici de criptare

Serialul dedicat criptării și decriptării mesajelor va continua cu prezentarea unor tehnici noi care măresc semnificativ securitatea mesajelor trimise.

Minimizare

Următorul număr al revistei va conține un articol în care se va prezenta un algoritm cu ajutorul căruia pot fi minimizate diferite funcții logice. Mai precis, se va prezenta metoda Quinne-McCluskey, una dintre cele mai eficiente metode de a realiza astfel de operații. Articolul începe prin prezentarea unor noțiuni teoretice referitoare la funcțiile logice, urmând ca apoi să fie prezentați pașii unui algoritm care poate fi folosit pentru a determina expresii cât mai simple pentru diferite funcții logice.

Might and Magic IX

Might and Magic este o serie înrudită cu Heroes of Might and Magic, cele două familii de jocuri fiind produse de aceeași companie. Universul jocurilor este același, acțiunea având loc pe aceleași meleaguri. Diferența esențială dintre cele două jocuri este genul în care se încadrează: în timp ce jocurile din seria Heroes of Might and Magic sunt strategii bazate pe ture, cele din seria Might and Magic sunt RPG-uri (Role Playing Game). În numărul următor veți putea citi o scurtă prezentare a celui de-al nouălea joc din această serie.

Might and Magic IX

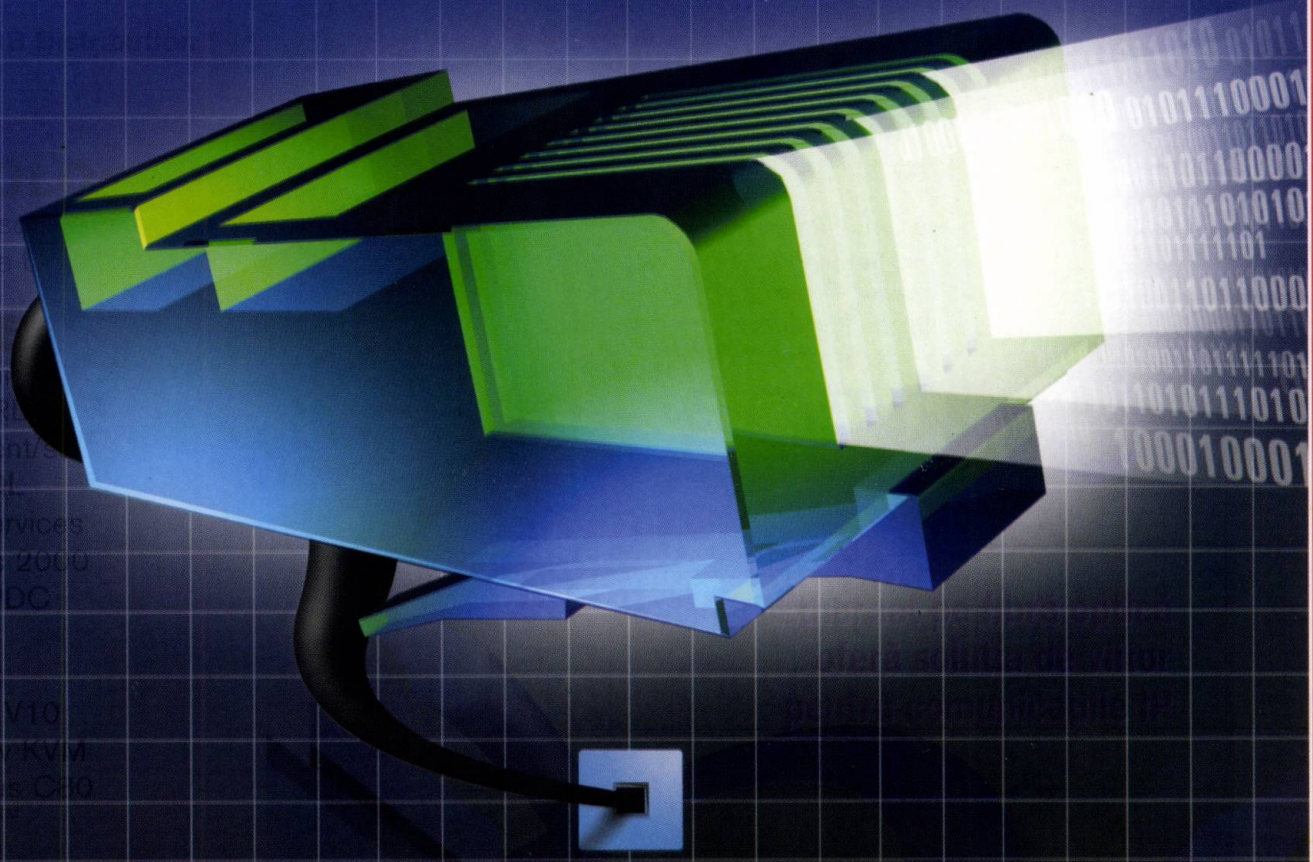
Bursele Agora

În numărul următor veți găsi prezentarea rezolvărilor celor trei probleme ale rundei a patra. De asemenea, va fi publicat clasamentul acestei etape, precum și clasamentul intermediar înaintea desfășurării ultimei etape a concursului nostru.

Rețele de calculatoare

Seria de articole dedicată rețelelor de calculatoare va continua cu prezentarea standardelor care stau la baza funcționării rețelelor locale clasice.

NETREPORT



Conectat la viitorul digital

AGORA MEDIA: Str. Tudor Vladimirescu nr. 63/9, Tg. Mureș, CP 230/1, 4300
Tel: 065-166 516, Fax: 065-166 290, e-mail: office@agora.ro

www.netreport.ro

Agora MEDIA

Publicarea acestei reviste este posibilă datorită sponsorizărilor acordate de următoarele firme care s-au decis să sprijine învățământul românesc de informatică

SPONSORI DE PLATINĂ

COMPAQ

SPONSORI DE AUR

FORTE COMPUTERS
Flexibility. Made in Romania

ASTRAL TELECOM

SPONSORI DE ARGINT

s&t
IT SOLUTIONS & SERVICES

hp
invent

ERICSSON

Magua

M.G.T.
EDUCATIONAL

a&C
INTERNATIONAL S.A.
the CAD-CAMpany
autodesk
authorized distributor

ALCATEL

*Rugăm companiile care doresc să se alăture acestui efort să ne contacteze prin e-mail la adresa office@ginfo.ro sau la numărul de telefon: 065-166.516

gazeta de informatica

Revista viitorilor specialiști în tehnologia informației

Aprilie 2002
Vol. 12/4

Lei 20.000

din sumarul numărului:

Actual

Profil Microsoft®

enjoy

Caracterizarea sistemelor nesecvențiale • Rețele Ethernet
Magazinul de jocuri • Tehnici de criptare • Teac CD-W540E
Minimizarea funcțiilor logice • Rețele de calculatoare

GHIDUL TĂU ÎN LUMEA TEHNOLOGIILOR

PC MAGAZINE ROMÂNIA

de reținut!

CONCURS DE FOTOGRAFIE DIGITALĂ

HP România –
PC Magazine România

1 martie - 24 mai 2002

www.hp.com.ro, www.pcmagazine.ro

Misiunea noastră este de a testa și evalua produse și servicii din domeniul tehnologiei informației și comunicațiilor și de a prezenta rezultatele cu onestitate și obiectivitate

AGORA MEDIA Str. C-tin Rădulescu Motru nr. 13, etaj 4, apt. 42, sect. 4 (CP 94/49), București
Tel: 01-3309282, Fax: 01-3309285

www.pcmagazine.ro

Agora MEDIA

Vol. 12, Nr. 4 /2002

gazeta de informatică

Redacția:

Director științific:
prof. dr. *Horia Georgescu*
Redactor șef:
lect. univ. *Clara Ionescu*
Redactor șef adjunct:
Mihai Scorțaru
Redactori:
prof. *Eugen Ionescu*
Carmen Șușman
Claudiu Soroiu
Au mai colaborat:
Vasile Airinei (Sebeș)
Oana Georgescu (Brașov)
Mihai Stroe (București)
Anca Vasilescu (Brașov)
Tehnoredactare:
Claudiu Soroiu
Coperta și grafica:
Andrei Pașa
Adresa pentru corespondență:
Computer Libris Agora,
Universității 8/8,
3400, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 064/192.422
065/166.516
Fax: 065/166.290
http://www.ginfo.ro
ISSN: 1222-7129

Tiparul: *Master Druck* Târgu Mureș
tel: 065/162.359

Editor: Agora Media SRL
Agora Media SRL, editor al publicațiilor "*eWeek România*", "*PC Magazine România*", "*NetReport*" și "*Gazeta de Informatică*", este membru al Biroului Român pentru Auditarea Tirajelor.
str. T. Vladimirescu nr. 63/9
4300 Târgu-Mureș
Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director tehnic:
Szabó László
Vânzare reclamă:
Mihai Bucuroiu
Bogdan Manu
Denis Mărioara
Marius Comărniceanu
Reprezentanța București:
Director: *Dana Pastramă*
Tel: 01/330.92.81
Fax: 01/330.92.85

Sumar

actual.. PAG. 5

Produsele lunii
Teac CD-W540E
Windows XP...
în română
Might and Magic IX
Magazinul de JOCURI

probleme.. PAG. 9

Olimpiada JUDEȚEANĂ de Informatică

~ subiecte unice în ~
~ întreaga țara ~

concurs.. PAG. 13

Rezolvările problemelor celei de-a patra runde

Clasamentul general

focus.. PAG. 18

MINIMIZAREA funcțiilor logice

Vasile AIRINEI

serial.. PAG. 24

Caracterizarea sistemelor NESECVENȚIALE

Anca VASILESCU
Oana GEORGESCU

Tehnici de CRIPTARE

Claudiu SOROIU

tehno.. PAG. 34

Rețele ETHERNET

Mihai SCORȚARU

~Rețele de calculatoare~

profil.. PAG. 37

MICROSOFT

Carmen ȘUȘMAN

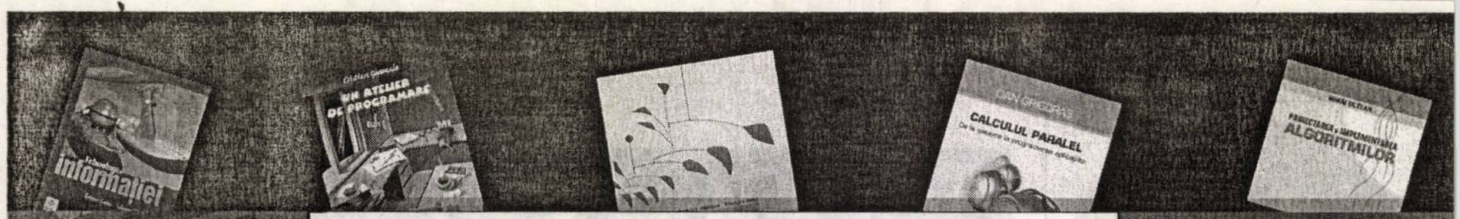
~ firme celebre ~

enjoy.. PAG. 39

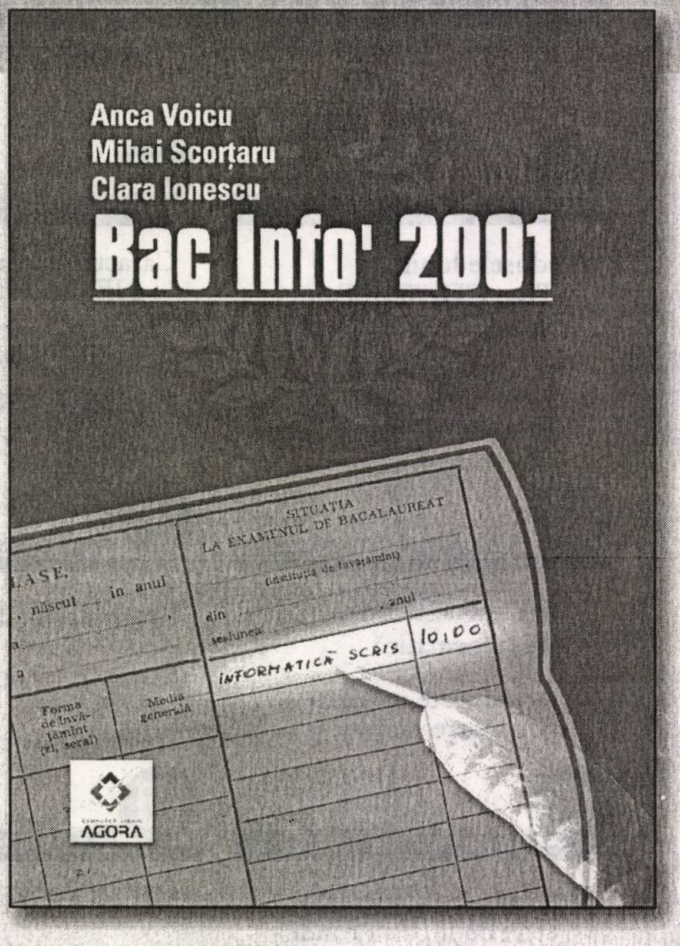
Heroes of Might & Magic IV

Mihai SCORȚARU

Sărbători fericite!



COMPUTER LIBRIS AGORA



Bac Info' 2001 conține 20 de variante de subiecte pentru Bacalaureat. Fiecare variantă conține câte patru subiecte. Primul dintre ele conține trei subpuncte care testează capacitățile candidaților de a recunoaște valoarea de adevăr a unei propoziții, de a determina efectul unui anumit algoritm și de a găsi o anumită greșeală într-un algoritm. Al doilea subiect conține doua subpuncte; primul testează deprinderea candidaților de a folosi limbajul pseudocod, iar al doilea cea de a folosi un limbaj de programare (Pascal sauC++). Al treilea subiect este o problemă pentru rezolvarea căreia este necesară folosirea unor structuri dinamice de date, iar al patrulea este o problemă pentru a cărei rezolvare trebuie folosită metoda backtracking.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

Homo LUDENS

Chiar dacă ne place sau nu, jocurile fac parte din viața noastră. Puțini sunt cei care pot susține că nu s-au bucurat deloc de momentele de relaxare pe care le oferă produsele de acest tip. Mulți pierd ore întregi în fața ecranului fără a-și da seama de trecerea timpului.

Dacă ați răsfoit puțin paginile acestui număr al revistei sau ați aruncat o privire asupra cuprinsului, ați putut observa că este dedicată, în mare parte, jocurilor.

Pe lângă obișnuita pagină în care prezentăm pe scurt un joc, puteți găsi nouă pagini în care am realizat un *review* complet al unui joc. De asemenea, *site*-ul pe care îl prezentăm în această lună este dedicat jocurilor.

Nu cred că există multe persoane cărora nu le place să se joace. Chiar dacă unii consideră că jocurile reprezintă o pierdere de timp, cu siguranță au fost și ei surprinși de câteva ori încercând să aranjeze cărțile într-o anumită ordine sau să descopere amplasamentul minelor.

Jocurile au intrat în viața celor ce utilizează calculatoarele și vor dispărea destul de greu. Important este ca ele să nu ne domine viața. Cred că vă puteți aminti de prieteni sau cunoștințe care, întrebați de ce nu au dormit în noaptea precedentă, v-au răspuns că nu reușeau nici cum să cucerească baza zergilor sau că au vrut să elimine teroriști timp de câteva minute și distracția s-a prelungit până dimineața etc.

Cu siguranță există multe aspecte negative, pierderea noțiunii timpului fiind doar unul dintre ele. În ciuda tuturor avertismentelor, cei pasionați nu pot renunța la "ritualul" din fiecare zi. Chiar dacă li se spune că trebuie să facă pauze de cel puțin zece minute la fiecare oră, importanța misiunilor în desfășurare este mult mai mare decât cuvintele auzite anterior.

Evident, pot apărea probleme de vedere, se pierde capacitatea de concentrare, oboseala este la ordinea zilei; dar acestea sunt probleme minore... "Mult mai importantă" este victoria asupra hoardelor inamice de la ora patru dimineața.

Îmi aduc aminte de o discuție pe un forum, referitoare la jocul *Heroes of Might and Magic*. Cel mai interesant mesaj suna aproximativ astfel: *Vă dau un sfat! Nu vă învățați tatăl să joace Heroes că nu vă mai puteți atinge de calculator. Am privit șocat timp de treizeci și opt de ore cum mișca entuziasmat mouse-ul pe ecran și nu se oprea nici măcar pentru a mânca.*

Unii își dau seama de efectele negative pe care le au jocurile asupra lor. Alții continuă fiindcă în fiecare lună apar cel puțin douăzeci de jocuri noi dintre care cel puțin cinci sunt foarte interesante și, bineînțeles, toate trebuie încercate.

O problemă delicată este cea a modului în care sunt achiziționate jocurile. Cu siguranță știți că a realiza copii ale copiilor sau ale originalelor reprezintă infracțiuni. Nu vă voi mai spune ce se poate întâmpla dacă sunteți "prinși" cu produse *software* deținute ilegal fiindcă veți zâmbi cu superioritate și veți spune ceva de genul: *"Nu poate să mă prindă nimeni!"*. Din nefericire, în majoritatea cazurilor aveți dreptate.

În dorința lor evidentă de a crea produse cât mai atractive, producătorii au reușit să creeze lumi virtuale care prezintă un pericol chiar prin faptul că ne amăgesc cu legi și decoruri, personaje și talente inexistente în lumea reală.

Am criticat destul jocurile fără a aminti și aspectele pozitive. Chiar dacă "inamicii" jocurilor nu vor să recunoască, acestea există și sunt destul de importante.

Utilitatea jocurilor logice este evidentă; nimeni nu poate susține că rezolvarea problemelor logice, chiar dacă acestea sunt transpuse pe calculator, are mai multe efecte negative decât pozitive.

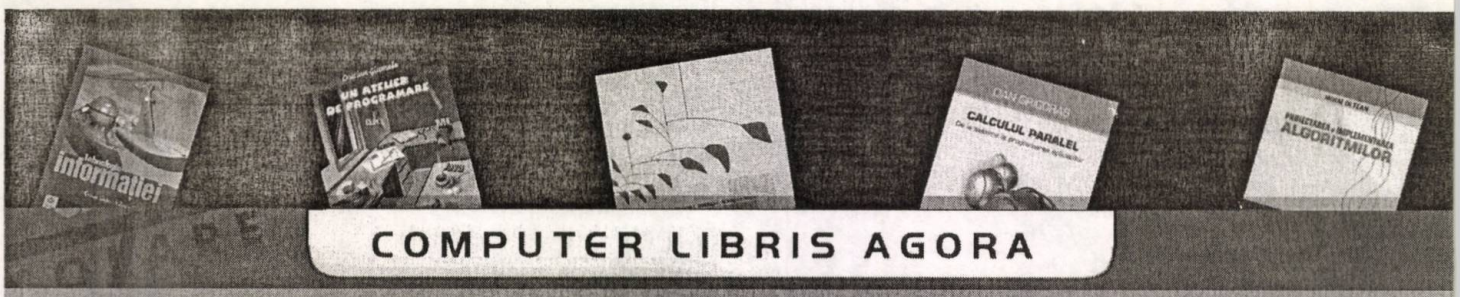
Jocurile de strategie, după cum sugerează denumirea, implică folosirea unor strategii. Construirea strategiilor necesită o gândire algoritmică bine pusă la punct; aceasta se dezvoltă, chiar dacă jucătorul nu este conștient de acest lucru, pe măsură ce acesta devine "expert" în jocurile de acest tip.

Jocurile de aventură sunt de fapt, un fel de jocuri logice îmbrăcate în "haine" mai frumoase. Mulți nu vor fi de acord cu această categorisire și probabil este cam exagerată...

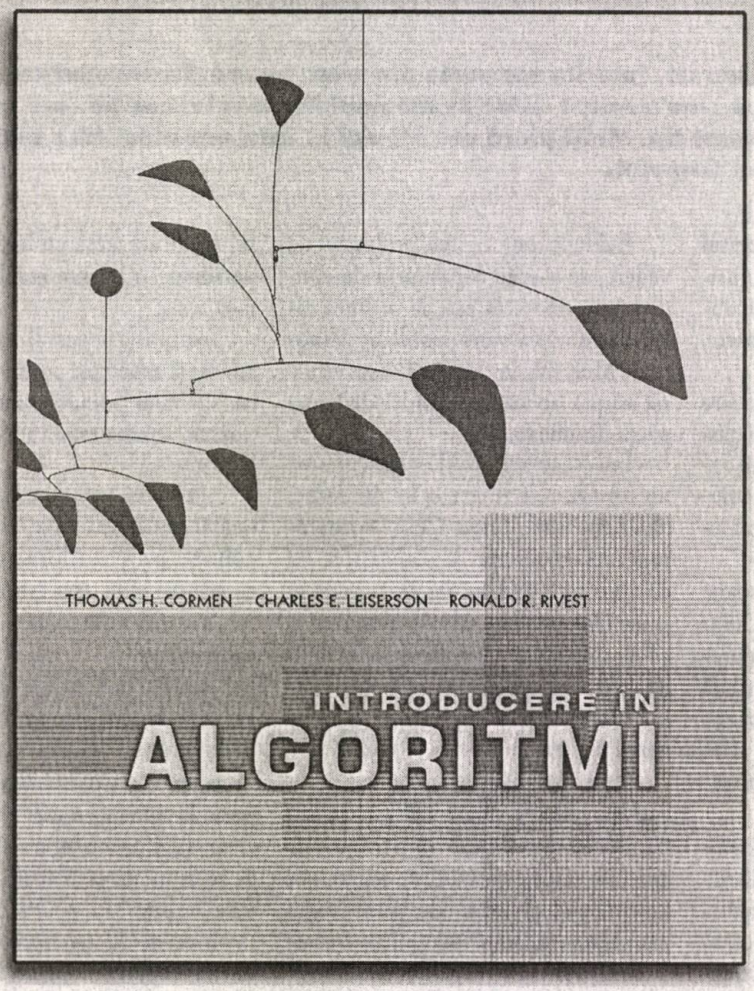
Jocurile de acțiune sunt cele mai "gustate" în acest moment. Se spune că ajută la îmbunătățirea vitezei de reacție. Tot ce se poate! Personal, consider că acestea reprezintă, cu adevărat, o pierdere de vreme.

Nu pot să ajung decât la concluzia logică ce se impune: ***jocurile pot avea efecte benefice dacă nu se exagerează.*** Din nefericire, de obicei se exagerează...

Mihai Scorțaru



COMPUTER LIBRIS AGORA



Cartea se adresează, în primul rând, celor care studiază la nivel elementar sau academic algoritmi și structuri de date. Deoarece tratează aspecte teoretice și practice, legate de proiectarea algoritmilor, este la fel de binevenită elevilor, studenților, profesorilor și autodidacților.
Introducere în algoritmi acoperă atât metodele clasice cât și metodele moderne ca analiza amortizată și algoritmii paraleli.
Prezentarea matematică, riguroasă, este atent detaliată în așa fel încât să fie accesibilă tuturor cititorilor.
Capitolele încep cu elemente de bază, avansând spre subiecte mai complexe.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

Teac CD-W540E

La sfârșitul lunii martie 2002 a fost lansată cea mai rapidă unitate de inscripționare a CD-urilor produsă de compania Teac. Viteza de inscripționare este 40x, cea de reinscripționare este 12x, iar cea de citire este 48x. Chiar dacă există deja pe piață unități CD-RW cu performanțe asemănătoare, cea distribuită de Teac oferă diferite facilități suplimentare, motiv pentru care a devenit produsul acestei luni.

Introducere

Pe data de 28 martie 2002 compania *Teac* a lansat cea mai performantă unitate *CD-RW* pe care a realizat-o și anume modelul *CD-W540E*.

Pachetul în care este livrat conține dispozitivul *CD-W540E*, a cărui mască poate fi de culoare albă sau neagră, manuale de instalare și utilizare, un cablu *IDE/ATAPI*, un cablu audio digital, un compact disc inscripționabil, unul reinscripționabil și un *CD* care conține *software* necesar inscripționării. Manualele de instalare și utilizare sunt traduse, pe lângă limba engleză, în limbile franceză și spaniolă.

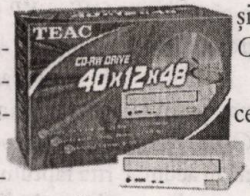
CD-ul cu *software* conține programele de instalare ale următoarelor produse:

- *Roxio Easy CD Creator* 5.2 care include pachetele *MusicCD*, *DataCD*, *Direct CD* și *CD Copier*; cu ajutorul pachetului *MusicCD* se pot realiza *CD*-uri audio din fișiere care pot fi stocate în format *wav*, *mp3*, *wma* etc.
- *PhotoImpression* 3.0 care poate fi folosit pentru editarea și inscripționarea imaginilor digitale;
- *Musicmatch Jukebox Basic* 7.0 care poate fi folosit pentru organizarea și inscripționarea pe *CD*-uri a fișierelor audio în format *MP3*.

Prețul unei unități *CD-RW Teac CD-W540E* era, în momentul lansării, de numai 180$.

Detalii tehnice

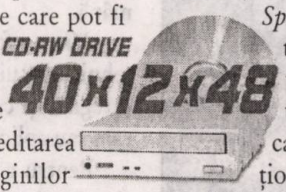
Dispozitivul este compatibil cu următoarele sisteme de operare realizate de *Microsoft*: *Windows* '98 *Second Edition*, *Windows Me*, *Windows NT* 4.0, *Windows* 2000 *Professional* și *Windows XP*.



Unitatea *CD-W540E* folosește cele mai noi tehnologii existente în acest domeniu la ora actuală.

Viteza maximă de inscripționare a dispozitivului este 40*x* (6 *MB/sec*), iar cea de reinscripționare este 12*x* (1,8 *MB/sec*).

Pentru inscripționare și reinscripționare, unitatea utilizează un *buffer* intern cu capacitatea de 8 *MB*, care folosește tehnologia *Buffer Underrun Prevention* pentru a putea continua procesul de scriere în condițiile unei variații foarte mari a ratei de transfer.



Cu ajutorul sistemului *Intelligent Speed Control*, unitatea detectează automat, la introducerea unui *CD* în unitate, viteza optimă la care acesta poate fi inscripționat, iar cu ajutorul sistemului *Fine Focus Control* se reglează automat intensitatea razei laser pentru o mai bună inscripționare sau citire a mediilor *CD*.

Produsul oferit de *Teac* este capabil să inscripționeze *CD*-uri care au diametrul de 8 *cm* sau de 12 *cm* și suportă mai multe formate de date, audio sau video.

Dispozitivul suportă următoarele moduri de inscripționare a mediilor *CD*: *Track at Once*, *Disk at Once*, *Multi-session* și *Incremental*.

Viteza de citire este 48*x* (7,2 *MB/sec*) pentru *CD*-urile care conțin date și 40*x* (6 *MB/ sec*) pentru *CD*-uri audio sau video.

Dispozitivul poate accesa o gamă largă de medii *CD-ROM*, *CD-R* și *CD-RW*.

Sistemul necesar

Unitatea *CD-W540E* poate fi instalată pe un calculator compatibil *IBM-PC* pe care rulează unul din sistemele de operare amintite anterior și care este echipat cu un procesor *Pentium* II (sau unul cu performanțe superioare) care are o frecvență de cel puțin 300 *MHz*.

Cantitatea minimă de memorie este de 64 *MB*, spațiul disponibil pe discul hard trebuie să aibă o dimensiune de cel puțin 1 *GB*, rata de transfer a discului trebuie să fie de cel puțin 1,2 *MB/sec*, iar viteza de acces de cel mult 1,2 *ms*.

Concluzii

Datorită performanțelor ridicate pe care le deține, noul produs oferit de compania *Teac*, este unul dintre cele mai importante dispozitive din domeniul inscriptoarelor în momentul de față.

Windows XP... în română

La câteva luni după apariția sistemului de operare Microsoft Windows XP Professional în limba engleză, Microsoft România a lansat pe piață versiunea cu interfață în limba română a acestuia. Utilizatorii români pot acum să interacționeze cu calculatorul într-un mod mult mai facil, ne mai având nevoie de cunoașterea limbii engleze.

Introducere

Microsoft România a anunțat pe data de 1 martie 2002 disponibilitatea sistemului de operare *Microsoft Windows XP Professional* în limba română.

Microsoft Windows XP Professional devine astfel primul sistem de operare, realizat de *Microsoft*, a cărui interfață a fost adaptată integral pentru limba română.

Folosind produsele *Windows XP Professional Romanian* și *Office XP Romanian*, utilizatorii români au acum posibilitatea de a interacționa cu calculatorul fără a mai fi necesare cunoștințe de limba engleză.

Descriere

Sistemul de operare *Windows XP* folosește tehnologia *Multilingual User Interface*, care oferă utilizatorilor posibilitatea de a avea instalate interfețe în mai multe limbi și de a comuta între ele.

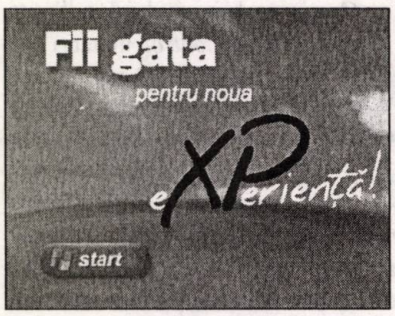
Din punct de vedere tehnic, sistemul de operare *Windows XP Professional Romanian* este versiunea în limba engleză la care s-au adăugat versiunile în limba română ale fișierelor de resurse. Șirurile de caractere care fac parte din interfață (titlurile ferestrelor, componentele meniurilor, fișierele de *help* etc.) au fost traduse și adaptate pentru limba română și au fost grupate în fișiere de resurse separate.

Traducerea sistemului de operare în diferite limbi se face mult mai ușor datorită utilizării tehnologiei *Multilingual User Interface*. Adaptarea interfeței *Windows XP Professional* pentru limba română a durat 3 luni de zile.

Utilizatorii pot rula aplicații care folosesc interfețe pentru diverse limbi (de exemplu, interfața *Windows XP* poate fi în limba română, în timp ce *Office XP* folosește interfața în limba franceză).

Compania *Microsoft* oferă acum suport tehnic pentru sistemul de operare *Windows XP* mult mai ușor datorită faptului că este vorba doar de versiunea în limba engleză a produsului la care s-a adăugat pachetul cu fișiere de resurse în limba română. Prin folosirea tehnologiei *Multilingual User Interface* se elimină incompatibilitățile datorate unui sistem realizat complet într-o anumită limbă.

Cu ajutorul acestei tehnologii, folosirea mai multor limbi devine mai ușoară.



Variante de achiziție

Interfața în limba română pentru *Windows XP Professional* este distribuită împreună cu sistemul de operare fie în varianta *OEM* (împreună cu un calculator nou), fie împreună cu pachetele care conțin *Windows XP Professional Upgrade* (pachete *Open Licensing*, *Open Subscription License*, *Software Assurance*, *Select* sau *Academic*).

Interfața în limba română pentru *Windows XP Professional* este disponibilă și în cadrul pachetului *MSDN* în variantele *Operating Systems*, *Professional*, *Enterprise* sau *Universal*.

Concluzii

Sistemul de operare *Microsoft Windows XP*, după cum se știe, este disponibil în două variante: *Home Edition* și *Professional*. Versiunea profesională include, pe lângă facilitățile versiunii *Home*, aplicații adiționale necesare rețelelor din mediile de afaceri.

Utilizatorii români care nu au nevoie de facilitățile suplimentare oferite de versiunea *Professional* sunt dezavantajați de faptul că varianta *Home Edition* nu are interfață în limba română.

Microsoft România oferă suport tehnic gratuit pentru toți utilizatorii înregistrați care dețin sisteme de operare sau aplicații de birou licențiate.

Pentru a adapta un produs la cerințele pieței din anumite zone geografice trebuie luați în considerare mai mulți factori printre care: volumul de vânzări ale unei versiuni anterioare ale produsului, rata pirateriei, returnarea investiției și argumente ce susțin creșterea vânzărilor pentru produsul respectiv dacă va fi disponibil în versiunea adaptată.

Might and Magic IX

Începând cu data de 31 martie 2002, una dintre cele mai cunoscute serii de RPG-uri are un nou membru: Might and Magic IX. În cadrul acestui număr vom realiza o scurtă prezentare a celui mai nou joc produs de compania 3DO.

Istorie

Seria *Might and Magic* a debutat în urmă cu aproape 20 de ani, când a fost lansat jocul *Might and Magic, Book* 1: *The Secret of the Inner Sanctum*. Acesta folosea grafica *EGA*, a fost realizat pe un calculator *Apple* II și a introdus o mulțime de noutăți în domeniul jocurilor de acest gen.

Primul joc s-a bucurat de un succes extraordinar, ceea ce a dus la apariția succesorului *Might and Magic* II: *Gates to Another World*. Chiar dacă nu au existat noutăți semnificative, jocul a fost foarte apreciat.

Lansarea jocului *Might and Magic* III: *Isles of Terra* a marcat un pas important din punct de vedere tehnologic. De această dată era folosită grafică *VGA* și era oferit suport pentru plăcile de sunet. Datorită acestor noutăți, jocul a devenit cel mai bun *RPG* al anului 1991 (premiu oferit de revista *Computer Gaming World*).

În anul 1992 a urmat *Might and Magic* IV: *Clouds of Xeen* care a fost unul dintre cele mai bine vândute jocuri ale momentului.

La puțin timp după acesta, a fost lansat *Might and Magic* V: *Darkside of Xeen*. Acest joc nu s-a ridicat la așteptările utilizatorilor și, datorită faptului că realizarea jocurilor de acest tip implica eforturi foarte mari, seria a fost întreruptă până în 1998.

Revenirea genului *RPG* a fost inițiată de *Might and Magic* VI: *The Mandate of Heaven*. Majoritatea producătorilor de jocuri ocoleau acest gen datorită faptului că dezvoltarea dura mai mulți ani și, în momentul lansării, jocurile erau depășite din punct de vedere tehnologic. La acest joc s-a lucrat timp de trei ani și, deși grafica lăsa de dorit, jocul a fost foarte apreciat. În scurt timp, genul *RPG* a intrat din nou în atenția producătorilor și acum există o mulțime de jocuri de acest tip.

A urmat în scurt timp *Might and Magic* VII: *For Blood and Honor*, un joc care avea o grafică îmbunătățită, dar era depășit de jocurile produse de companiile rivale.

Might and Magic VIII: *Day of the Destroyer* a dezamăgit datorită calității graficii; motorul grafic era cel folosit la crearea predecesorului.

A urmat o nouă pauză de câțiva ani dar, în martie 2002, a apărut ultimul joc al seriei: *Might and Magic* IX.

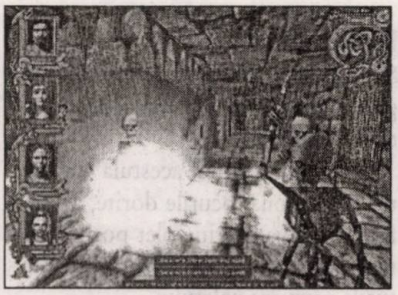
De-a lungul existenței seriei, cele opt jocuri *Might and Magic* s-au vândut în peste cinci milioane de exemplare.

Caracteristici

Might and Magic IX este primul joc al seriei care folosește grafica *3D*. Această noutate permite crearea unei lumi mai realiste și îmbunătățește substanțial calitatea jocului.

Inteligența artificială a fost mult îmbunătățită, monștrii inamici având diverse strategii de luptă.

A fost modificat și sistemul de conversație cu personajele neutre. Replicile pe care acestea le au la dispoziție sunt actualizate în funcție de acțiunile jucătorului.



Firul epic

Ar fi trebuit să fie doar un simplu transport de produse din *Chedian* spre *Mendossus*. Din nefericire, a apărut o furtună și nava a naufragiat pe o insulă (*Isle of Ashes*). Aici începe aventura voastră...

Între timp ținutul natal este atacat... Vă descoperiți destinul atunci când aflați că sunteți singura speranță a celor din *Chedian*. Va trebui să unificați cele șase clanuri pentru a avea șanse de supraviețuire.

Soarta pașnicilor locuitori ai ținuturilor natale se află în mâinile voastre...

Încheiere

Might and Magic IX promite să relanseze această serie de jocuri. Producătorii au încercat să mulțumească toți jucători. Dumneavoastră veți decide în ce măsură au reușit.

Might and Magic IX

Magazinul de JOCURI

Puțini jucători folosesc în mod legal produsele software de acest tip. Aceștia pot acum să beneficieze de prețurile speciale pentru sud-estul Europei. Prin intermediul site-ului www.gameshop.ro puteți achiziționa cele mai noi jocuri, prețurile fiind mai mici decât v-ați aștepta...

Prezentare

Site-ul **www.gameshop.ro** este un magazin virtual dedicat aproape în exclusivitate jocurilor.

Prin intermediul acestuia veți putea achiziționa jocurile dorite, acestea fiind expediate prin colet poștal într-un timp foarte scurt. Plata se va efectua la primirea coletului.

Fiecărui joc disponibil îi este dedicată o pagină care cuprinde detalii referitoare la preț, precum și o scurtă descriere a jocului. De asemenea, este prezent un *link* spre *site*-ul oficial dedicat jocului. Vizitatorii pot să își facă publică părerea referitoare la fiecare joc, mesajele lor fiind publicate în pagina dedicată jocului. De asemenea, a fost implementat un sistem de punctare a jocurilor prin intermediul căruia vizitatorii pot acorda note fiecărui joc.

Efectuarea comenzilor

Pentru a putea comanda jocuri, utilizatorii trebuie să își creeze un cont. Pentru aceasta ei vor completa un formular cu datele personale, acestea fiind necesare pentru ca produsele să poată fi expediate.

După selectarea jocurilor dorite, cumpărătorii vor introduce numele de utilizator și parola, după care vor confirma comanda. După puțin timp, ei vor fi contactați telefonic pentru a confirma din nou comanda. Această măsură supimentară de securitate este folosită pentru a evita comenzile fictive.

Servicii oferite

Pe lângă jocurile disponibile care vor fi expediate imediat, utilizatorii pot cumpăra diverse jocuri care nu sunt disponibile încă pe piață. Ei vor plăti în avans prețul pentru jocurile respective, iar acestea vor fi expediate în ziua lansării oficiale a jocului. Astfel, cei care doresc să fie primii posesori ai anumitor jocuri, vor putea să se asigure că vor avea la dispoziție produsul dorit cât mai curând posibil. De asemenea, cei care folosesc această facilitate vor primi diferite cadouri promoționale.

Cei care vor comanda produse cu valoarea mai mare decât o anumită sumă, vor primi și ei diferite cadouri. În fiecare lună există o promoție de acest tip, cadourile constând în dispozitive *hardware* sau alte jocuri care vor fi primite gratuit.

G@MESHOP.RO

Prețuri

Pentru a achiziționa jocurile aveți la dispoziție un coș de cumpărături virtual. În orice moment veți putea elimina sau adăuga jocuri în acest coș. De asemenea, veți cunoaște tot timpul valoarea produselor din coș.

În paginile dedicate jocurilor sunt scrise prețurile corespunzătoare, exprimate în *USD* (acestea nu includ *TVA*). Valoarea totală a cumpărăturilor este exprimată în lei și conține *TVA*.

În momentul efectuării comenzii veți cunoaște valoarea în lei a produselor, precum și taxele poștale corespunzătoare. La începutul lunii aprilie valoarea taxelor poștale era de 75.000 lei, indiferent de numărul jocurilor achiziționate sau de localitatea în care doriți să primiți coletul.

Site-ul vă pune la dispoziție o listă cu cele mai bine vândute jocuri, precum și una care conține ultimele jocuri vândute.

Promovare

Site-ul este promovat prin intermediul diferitelor reclame prezente pe alte *site*-uri. Campania de promovare nu este deloc agresivă, ea limitându-se la afișarea unor *banner*-e publicitare.

Pentru a atrage cumpărătorii este organizat în fiecare lună un concurs cu premii. Pentru înscrierea la concurs este suficient să precizați numele de utilizator și parola. Premiile vor fi oferite prin tragere la sorți la sfârșitul fiecărei luni.

Concluzii

Site-ul **www.gameshop.ro** este destul de bine realizat, oferind toate informațiile de care au nevoie cei care doresc să achiziționeze jocuri. Ofertele speciale sunt destul de atrăgătoare; de exemplu puteți achiziționa jocuri celebre apărute în urmă cu doi sau trei ani la prețuri foarte mici. Celebrul *Heroes of Might and Magic* III: *Restoration of Erathia* costă doar 6$; există numeroase alte jocuri mai vechi care pot fi achiziționate la acest preț.

Olimpiada JUDEȚEANĂ de informatică

Pentru prima dată, în acest an olimpiada județeană s-a desfășurat folosindu-se aceleași subiecte în întreaga țară. În continuare vă prezentăm enunțurile celor șase probleme cu care s-au confruntat concurenții.

P040201: Poarta

(clasa a IX-a)

Se consideră harta universului ca fiind o matrice cu 250 de linii și 250 de coloane. În fiecare celulă se găsește o așa numită *poartă stelară*, iar în anumite celule se găsesc echipaje ale *porții stelare*. La fiecare pas, un echipaj se poate deplasa din locul în care se află în oricare alt loc în care se găsește o a doua poartă, în cazul nostru în orice altă poziție din matrice. Nu se permite situarea simultană a mai mult de un echipaj într-o celulă. La un moment dat un singur echipaj se poate deplasa de la o poartă stelară la alta.

Dându-se un număr p de echipaje, pentru fiecare echipaj fiind precizate poziția inițială și poziția finală, determinați numărul minim de deplasări necesare pentru ca toate echipajele să ajungă din poziția inițială în cea finală.

Date de intrare

Fișierul de intrare **POARTA.IN** are următorul format:

- prima linie conține un număr natural p, care reprezintă numărul de echipaje;
- următoarele p linii conțin câte 4 numere naturale separate printr-un singur spațiu; primele două reprezintă coordonatele poziției inițiale (linie și coloană) a unui echipaj, iar următoarele două reprezintă coordonatele poziției finale ale aceluiași echipaj.

Date de ieșire

Pe prima linie a fișierului text **POARTA.OUT** se scrie un singur număr, reprezentând numărul minim de deplasări necesare.

Restricții și precizări

- $1 < p < 5000$;
- coordonatele pozițiilor inițiale și finale ale echipajelor sunt numere naturale din intervalul [1, 250];
- pozițiile inițiale ale celor p echipaje sunt distincte două cîte două;
- pozițiile finale ale celor p echipaje sunt distincte două câte două.

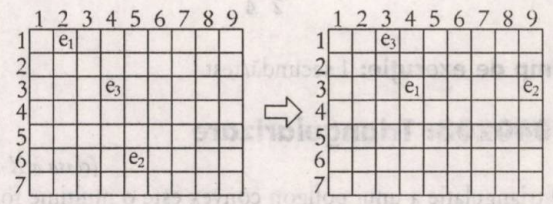
Exemplu

POARTA.IN

```
3
1 2 3 4
6 5 3 9
3 4 1 2
```

POARTA.OUT

```
4
```



Timp de execuție: 1 secundă/test

P040202: Mouse

(clasa a IX-a)

Un experiment urmărește comportarea unui șoricel pus într-o cutie dreptunghiulară, împărțită în $m \times n$ cămăruțe egale de formă pătrată. Fiecare cămăruță conține o anumită cantitate de hrană. Șoricelul trebuie să pornească din colțul (1, 1) al cutiei și să ajungă în colțul opus, mâncând cât mai multă hrană. El poate trece dintr-o cameră în una alăturată (două camere sunt alăturate dacă au un perete comun), mănâncă toată hrana din cămăruță atunci când intră și nu intră niciodată într-o cameră fără hrană. Stabiliți care este cantitatea maximă de hrană pe care o poate mânca și traseul pe care îl poate urma pentru a culege această cantitate maximă.

Date de intrare

Fișierul de intrare **MOUSE.IN** conține pe prima linie două numere m și n, reprezentând numărul de linii, respectiv

numărul de coloane ale cutiei, iar pe următoarele m linii cele $m \times n$ numere reprezentând cantitatea de hrană existentă în fiecare cămăruță, câte n numere pe fiecare linie, separate prin spații.

Date de ieșire

În fișierul de ieșire **MOUSE.OUT** se vor scrie pe prima linie două numere separate printr-un singur spațiu care reprezintă numărul de cămăruțe vizitate, respectiv cantitatea de hrană maximă culeasă. Pe următoarele linii se va scrie un traseu posibil pentru cantitatea dată, sub formă de perechi de numere (linie și coloană) separate printr-un singur spațiu, începând cu perechea (1, 1) și terminând cu perechea (m, n).

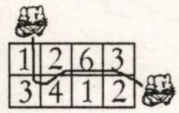
Restricție

- toate valorile din fișierul de intrare sunt numere naturale cuprinse între 1 și 100.

Exemplu

MOUSE.IN

```
2 4
1 2 6 3
3 4 1 2
```



MOUSE.OUT

```
7 21
1 1
2 1
2 2
1 2
1 3
1 4
2 4
```

Timp de execuție: 1 secundă/test

P040203: Triangularizare

(clasa a X-a)

O triangulație a unui poligon convex este o mulțime formată din diagonale ale poligonului care nu se intersectează în interiorul poligonului ci numai în vârfuri și care împart toată suprafața poligonului în triunghiuri.

Fiind dat un poligon cu n vârfuri notate cu 1, 2, ..., n, să se genereze toate triangulațiile distincte ale poligonului. Două triangulații sunt distincte dacă diferă prin cel puțin o diagonală.

Date de intrare

Fișierul **TRIANG.IN** conține o singură linie pe care se află un număr întreg pozitiv reprezentând valoarea lui n.

Date de ieșire

Fișierul de ieșire **TRIANG.OUT** va conține:

- pe prima linie, numărul de triangulații distincte;
- pe fiecare din următoarele linii câte o triangulație descrisă prin diagonalele care o compun. O diagonală va fi precizată prin două numere reprezentând cele două vârfuri care o definesc; cele două numere care definesc o diagonală vor fi separate printr-un singur spațiu, iar două diagonale dintr-o triangulație vor fi separate printr-un singur spațiu.

Restricții și precizări

- $N \leq 11$.

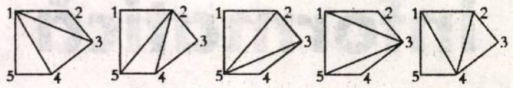
Exemplu

TRIANG.IN

```
5
```

TRIANG.OUT

```
5
1 3 1 4
2 4 2 5
5 2 5 3
3 5 3 1
4 2 1 4
```



Timp de execuție: 3 secunde/test

P030204: Cod

(clasa a X-a)

Principala misiune a unei expediții științifice este de a studia evoluția vieții pe o planetă nou descoperită. În urma studiilor efectuate, cercetătorii au asociat fiecărui organism viu descoperit pe acea planetă un cod caracteristic.

De asemenea, cercetătorii au observat că pentru orice organism viu de pe planetă, codurile caracteristice ale strămoșilor săi pe scara evoluției se pot obține prin ștergerea unor cifre din codul caracteristic al organismului respectiv, iar un organism este cu atât mai evoluat cu cât codul său caracteristic are o valoare mai mare.

Date de intrare

Fișierul de intrare **COD.IN** conține două linii. Pe prima linie se află codul caracteristic n al primului organism, iar pe cea de-a doua linie se află codul caracteristic m al celui de-al doilea organism.

Date de ieșire

Fișierul de ieșire **COD.OUT** va conține o singură linie pe care se va afla codul p al celui mai evoluat strămoș comun al lui n și m.

Restricții și precizări

- codul caracteristic este un număr natural care are cel mult 200 de cifre.

Exemplu

COD.IN

```
7145
847835
```

COD.OUT

```
75
```

Timp de execuție: 1 secundă/test

P030205: Nunta

(clasele a XI-a și a XII-a)

În fața palatului *Prințesei Mofturoase* se află n pețitori așezați la coadă, unul în spatele celuilalt. Fiecare poartă sub mantie un număr de pietre prețioase pe care dorește să le

ofere prințesei ca dar de nuntă. Pentru a nu semăna vrajbă în rândurile lor, prințesa a decis să-i determine ca n - 1 dintre ei să renunțe în chip pașnic, pețitorul rămas devenind alesul prințesei (indiferent de numărul de pietre prețioase deținute de acesta).

Doi pețitori vecini la coadă se pot înțelege între ei astfel: cel care are mai puține pietre prețioase pleacă de la coadă primind de la celălalt un număr de pietre astfel încât să plece acasă cu un număr dublu de pietre față de câte avea. Dacă doi pețitori au același număr de pietre, unul din ei (nu contează care) pleacă luând toate pietrele vecinului său.

Un pețitor se poate înțelege la un moment dat cu unul singur dintre cei doi vecini ai săi. După plecarea unui pețitor, toți cei din spatele lui avansează.

Fie p numărul de pietre prețioase pe care le are pețitorul care va deveni alesul prințesei. Se cer valorile distincte ale lui p la care se poate ajunge prin toate succesiunile de negocieri posibile.

Date de intrare

Fișierul de intrare **NUNTA.IN** conține două linii. Pe prima linie se află numărul n al pețitorilor, iar pe cea de-a doua linie se află n numere naturale separate printr-un singur spațiu și reprezintă numărul de pietre prețioase pe care le dețin pețitorii, în ordinea în care stau la coadă.

Date de ieșire

Fișierul de ieșire **NUNTA.OUT** va conține două linii. Pe prima linie se va scrie numărul m al valorilor distincte care pot fi obținute, iar pe cea de-a doua linie se vor scrie cele m valori ordonate crescător, separate printr-un singur spațiu.

Restricții și precizări

- $1 \le n \le 50$;
- numărul de pietre pe care îl deține fiecare pețitor este cuprins în intervalul [0, 20].

Exemplu

NUNTA.IN	NUNTA.OUT
4	3
1 4 2 6	1 3 5

Timp de execuție: 1 secundă/test

P030206: Urgența

(clasele a XI-a și a XII-a)

Autoritățile dintr-o zonă de munte intenționează să stabilească un plan de urgență, pentru a reacționa mai eficient la frecventele calamități naturale din zonă. În acest scop au identificat N puncte de interes strategic și le-au numerotat distinct de la 1 la N. Punctele de interes strategic sunt conectate prin M căi de acces având priorități în funcție de importanță. Între oricare două puncte de interes strategic există cel mult o cale de acces care poate fi parcursă în ambele sensuri și cel puțin un drum (format din una sau mai multe căi de acces) care le conectează.

În cazul unei calamități unele căi de acces pot fi temporar întrerupte, și astfel, între anumite puncte de interes nu mai există legătură. Ca urmare, pot rezulta mai multe grupuri de puncte, în așa fel încât, între oricare două puncte din același grup să existe măcar un drum și între oricare două puncte din grupuri diferite să nu existe drum.

Autoritățile estimează gravitatea unei calamități ca fiind suma priorităților căilor de acces distruse de aceasta și doresc să determine un scenariu de gravitate maximă, în care punctele de interes strategic să fie împărțite într-un număr de K grupuri.

Date de intrare

Fișierul de intrare **URGENTA.IN** are următoarea structură:

- pe prima linie se află trei numere întregi pozitive N, M, K separate printr-un singur spațiu;
- pe următoarele M linii se află câte trei numere întregi pozitive separate printr-un singur spațiu i, j, p care semnifică faptul că între punctele i și j există o cale de acces de prioritate p.

Date de ieșire

Fișierul de ieșire **URGENTA.OUT** va avea următoarea structură:

- pe prima linie se va scrie un număr *gravmax* care reprezintă gravitatea maximă;
- pe a doua linie se va scrie numărul C al căilor de acces întrerupte de calamitate;
- pe fiecare din următoarele c linii se vor afla două numere k și h separate printr-un singur spațiu și indică faptul că a fost întreruptă calea de acces dintre punctele k și h.

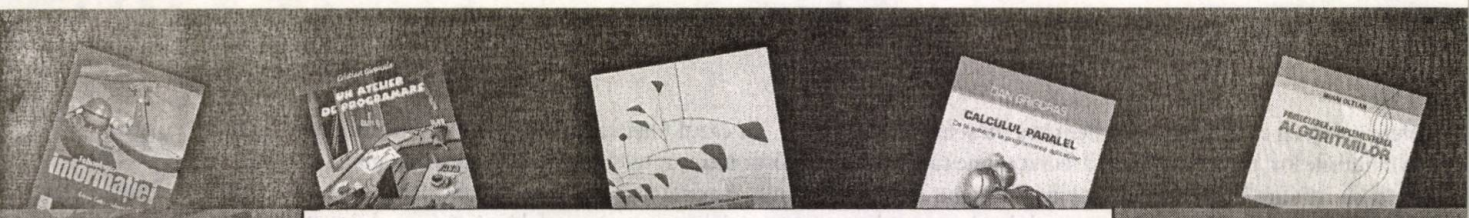
Restricții și precizări

- $0 < K \le N \le 256$;
- $2 \cdot N < M < 32385$;
- prioritățile căilor de acces sunt numere întregi, strict pozitive mai mici decât 256;
- un grup de puncte poate conține între 1 și N puncte;
- dacă există mai multe soluții, se va determina una singură.

Exemplu

URGENTA.IN	URGENTA.OUT
7 11 4	27
1 2 1	8
1 3 2	1 3
1 7 3	1 7
2 4 3	2 4
3 4 2	3 4
3 5 1	3 7
3 6 1	4 5
3 7 5	5 6
4 5 5	6 7
5 6 4	
6 7 3	

Timp de execuție: 1 secundă/test



COMPUTER LIBRIS AGORA

Calculul paralel. De la sisteme la aplicații tratează, într-un mod natural și structurat, cele mai importante aspecte ale proiectării aplicațiilor paralele, sau, cu un termen mai nou, de mare performanță. Paralelismul, care înseamnă executarea mai multor operații simultan, este o caracteristică a tuturor calculatoarelor actuale, inclusiv a celor personale. De aceea, cunoașterea principiilor și legilor care îl guvernează, ca și a metodologiilor pentru proiectarea aplicațiilor cele mai performante este o necesitate,dacă se dorește exploatarea optimă a resurselor sistemelor de calcul. Parametrii de performanță și metodologiile de proiectare și execuție optimă reprezintă subiectele care guvernează discursul științific al cărții de față.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS
AGORA

Bursele AGORA

A patra etapă a concursului de programare Bursele Agora a luat sfârșit. În cadrul acestui număr vă vom prezenta soluțiile problemelor de la această rundă, clasamentul etapei, precum și clasamentul general înaintea desfășurării ultimei etape.

A patra rundă

Problemele celei de-a patra runde au fost mai puțin dificile decât cele de la etapele anterioare. O dovadă în acest sens este faptul că patru dintre concurenți au reușit să obțină punctajul maxim: *Mugurel-Ionuț Andreica* din *București*, *Cosmin-Silvestru Negrușeri* din *Bistrița*, *Claudiu Raicu* din *Pitești* și *Paul Rusu* din *Cluj-Napoca*.

Chiar dacă doar cei patru au obținut punctajul maxim la problema *Arbori*, celelalte două probleme au fost rezolvate de un număr destul de mare de concurenți, deci puteau fi obținute destul de ușor punctaje de peste 200 de puncte.

Din nefericire s-au pierdut din nou puncte datorită unor erori banale cum ar fi folosirea *unit*-ului CRT, citirea datelor de intrare de la tastatură, afișarea pe ecran a datelor de ieșire etc. Unii concurenți au afișat diferite informații pe ecran ceea ce a dus la expirarea timpului de execuție în unele cazuri. Primii 20 clasați la această rundă sunt:

1. Mugurel-Ionuț Andreica, București	*300*
1. Cosmin-Suilvestru Negrușeri, Bistrița	*300*
1. Claudiu Raicu, Pitești	*300*
1. Paul Rusu, Cluj-Napoca	*300*
5. Radu Dondera, București	*288*
5. Maximilian Machedon, București	*288*
7. Radu Berinde, București	*261*
7. Tiberiu Dăneț, București	*261*
7. Octavian-Daniel Dumitran, București	*261*
10. Csaba Pătcaș, Oradea	*234*
11. Victor-Marius Costan, București	*209*
12. Marius-Dorin Moraru, Buftea	*192*
13. Andrei Homescu, Târgu-Jiu	*184*
14. Liviu-Cosmin Andreicuț, Bacău	*183*
15. Johannes Slotta, Germania	*182*
16. Radu-Andrei Ștefan, Brașov	*172*
17. Andrei David, Bacău	*168*
18. Andrei Vancea, Cluj-Napoca	*164*
19. Victoria Tarasova, Ucraina	*158*
20. Flaviu Roman, Cluj-Napoca	*154*

Datele de test folosite pentru testare, precum și variante ale ieșirilor corecte, pot fi descărcate folosind *link*-urile de pe paginile dedicate celor trei probleme. Adresele acestor pagini sunt:

- **www.ginfo.ro/concurs/arbori.shtml**;
- **www.ginfo.ro/concurs/atlantis.shtml**;
- **www.ginfo.ro/concurs/celule.shtml**.

Tot pe *site* veți putea găsi clasamanetul integral al rundei a patra (**www.ginfo.ro/concurs/runda4/top.shtml**), precum și clasamentul general (**www.ginfo.ro/concurs/top.shtml**).

Problemele *Atlantis* și *Celule* sunt destul de cunoscute, ele fiind prezente la diferite concusuri care s-au desfășurat în ultimii ani. Acesta este unul dintre motivele pentru care punctajele obținute de concurenți la aceste două probleme au fost relativ mari.

Sistemul de evaluare

Anunțăm faptul că problemele ultimei runde vor fi corectate folosind un calculator dotat cu un procesor *AMD Athlon XP* 1600+. Celelalte caracteristici ale sistemului de evaluare rămân neschimbate.

La mulți ani!

Vlad Ilie, Craiova - 4 IV 1984
Alexandra Constantin, Ploiești - 6 IV 1984
Gheorghe Fan, Ștefești - 7 IV 1984
Andrei Benea, Focșani - 10 IV 1983
Florin Gheț, Focșani - 11 IV 1982
Silviu-Georgian Aprozeanu, Botoșani - 14 IV 1983
Mihai Stoicoi, Orăștie - 14 IV 1984
Omar Choudary, București - 15 IV 1984
Ioana-Violeta Brutaru, Hunedoara - 20 IV 1983
Radu-Teodor Băluță, Târgu-Jiu - 22 IV 1983
Bogdan Nicolae, Sibiu - 23 IV 1983
Alexandru Roșiu, București - 28 IV 1982

Runda 4, problema 1

ARBORI

Această problemă a fost rezolvată corect doar de patru dintre participanții la a patra rundă a concursului *Bursele Agora*, ea fiind cea mai dificilă problemă propusă spre rezolvare la această etapă.

Algoritmul care trebuia aplicat pentru rezolvarea problemei constă din doi pași. La început se va determina costul transformării unui arbore din curte în fiecare dintre arborii din poză. Acest algoritm este destul de simplu, el neridicând dificultăți deosebite. La al doilea pas se va aplica un algoritm de determinare a unui cuplaj maxim de cost minim pentru un graf construit pe baza costurilor determinate la primul pas. Acest algoritm este mai dificil, dar el este prezentat în majoritatea lucrărilor care prezintă teoria grafurilor. Așadar, chiar dacă algoritmul nu era cunoscut, nu era foarte dificil de găsit și de implementat.

Primul pas care trebuie realizat în rezolvarea problemei este calcularea, pentru fiecare pereche de forma *arbore_din_curte - arbore_din_poză*, a costului (numărul de crengi tăiate) necesar transformării.

Pentru aceasta se poate folosi o funcție recursivă care returnează valoarea -1 dacă transformarea nu este posibilă (arborele din poză conține crengi în anumite poziții în care arborele din grădină nu conține crengi) sau costul cerut, calculat în funcție de costurile necesare transformării subarborilor stâng și drept (dacă oricare dintre ei este vid pentru arborele din poză și nevid pentru cel din curte, se adaugă valoarea 1 la costul corespunzător celuilalt subarbore și nu se mai efectuează apel recursiv pe ramura respectivă).

Este esențial să începem cu această operație, deoarece nu se dorește recalcularea acestor costuri de fiecare dată când avem nevoie de ele.

Rezultatul va fi o matrice C; aceasta poate fi privită ca fiind un graf bipartit; partițiile nodurilor sunt "*arborii din curte*" și "*arborii din poză*". Problema cere să asociem în mod biunivoc cât mai multor noduri din prima partiție, câte un nod din a doua partiție, astfel încât costul total să fie minim.

Așadar, problema se reduce la aplicarea unui algoritm pentru determinarea unui ***cuplaj maximal de cost minim***, care va fi rezolvată conform unor algoritmi foarte cunoscuți.

Graful se poate transforma într-o rețea de transport prin introducerea unui *nod sursă* (conectat cu arborii din curte) și a unui *nod destinație* (conectat cu arborii din poză).

Au rezolvat corect:

MUGUREL-IONUȚ ANDREICA, BUCUREȘTI
COSMIN-SILVESTRU NEGRUȘERI, BISTRIȚA
CLAUDIU RAICU, PITEȘTI
PAUL RUSU, CLUJ-NAPOCA

Toate arcele din graful bipartit inițial au capacitatea 1, iar sensul este dinspre sursă spre destinație. Costul arcelor introduse de nodurile sursă și destinație este 0.

Trebuie să determinăm fluxul maxim de cost minim în această rețea. După cum se știe, fluxul maxim se determină găsind repetat *drumuri de creștere*, de la sursă la destinație. Un drum de creștere poate să conțină un arc (i, j), dacă nu există flux pe el, sau dacă există flux pe arcul (j, i) - deci se parcurge arcul (j, i) în sens invers.

Se caută un astfel de drum de la sursă la destinație, se actualizează fluxul pe arcele care formează drumul (devine 1 dacă valoarea a fost 0 și 0 dacă valoarea a fost 1), se caută un nou drum de creștere etc.

Operația se repetă până în momentul în care nu mai există nici un drum de creștere. Inițial se va considera că fluxurile corespunzătoare tuturor arcelor din graf au valoarea 0.

Pot exista mai multe drumuri de creștere la un moment dat. În această situație va trebui ales *drumul de creștere de cost minim*.

Costul unui drum de creștere este dat de suma costurilor arcelor componente.

Costul unui arc este pozitiv dacă valoarea fluxului de pe acel arc este 0 și negativ în cazul în care fluxul de pe acel arc are valoarea 1. Valoarea absolută a acestui flux este dată de costul determinat la pasul anterior.

În continuare se va construi graful format din arcele corespunzătoare (cele pentru care valoarea fluxului este 1 sunt inversate și vor avea cost negativ; celelalte arce sunt păstrate fără a fi modificate). În acest graf, pe baza algoritmului *Belmann-Ford*, se va determina un astfel de drum de cost minim.

Algoritmul *Belmann-Ford* va funcționa corect deoarece graful nu va conține cicluri cu cost negativ. În final, se va obține un flux maxim de cost minim; arcele pe care fluxul are valoarea 1 constituie soluția problemei.

Codul sursă al programului este disponibil pentru *download* la adresa **www.ginfo.ro/concurs/arbori.shtml**.

GInfo Top 50

1. *Mugurel-Ionuț Andreica, București* 1139
2. *Silvestru-Cosmin Negrușeri, Bistrița* 1090
3. *Tiberiu Dăneț, București* 918
4. *Octavian-Daniel Dumitran, București* 902
5. *Maximilian Machedon, București* 849
6. *Radu Dondera, București* 822
7. *Adrian Balașko, Zalău* 808
8. *Claudiu Raicu, Pitești* 807
9. *Radu Berinde, București* 806
10. *Csaba András, Oradea* 804
10. *Paul Rusu, Cluj-Napoca* 804
12. *Radu-Andrei Ștefan, Brașov* 771
13. *Bogdan Nicolae, Sibiu* 769
14. *Csaba Pătcaș, Oradea* 752
15. *Andrei Vancea, Cluj-Napoca* 739
16. *Adrian-Nicolae Cârcu, Bistrița* 714
17. *Ștefăniță Fechete, Suceava* 699
18. *Andrei Homescu, Târgu-Jiu* 670
19. *Liviu-Cosmin Andreicuț, Bacău* 652
20. *Ioan-Claudiu Gruia, București* 603
21. *Victor-Marius Costan, București* 584
22. *Emilian Miron, Bacău* 482
23. *Andrei Giurgiu, București* 463
24. *Ștefan Gheorghe, București* 416
25. *Alpár-Ferenc Perini, Sfântu Gheorghe* 405
26. *Marius-Dorin Moraru, Buftea* 388
27. *Victoria Tarasova, Ucraina* 383
28. *Slotta Johannes, Germania* 375
29. *Aurelian Marinescu, Brașov* 375
30. *Andrei David, Râmnicu Vâlcea* 371
31. *Ștefan Bucur, București* 367
32. *Flaviu Pașca, Bistrița* 366
33. *Constantin Răuțu, Botoșani* 365
34. *Mihai-Daniel Cilidariu, Botoșani* 344
35. *Cristian Băicoianu, Ploiești* 335
36. *Liviu-Laurențiu Iacob, București* 332
37. *Bogdan-Vasile Hârjoc, Cugir* 324
38. *Mihail-Cosmin Pițu-Rada, Drobeta-Turnu Severin* 321
39. *Andrei Benea, Focșani* 313
40. *Sorin Otescu, Brașov* 310
41. *Adrian Airinei, Bacău* 306
42. *Mihai Pantelimon, București* 284
43. *Flaviu Roman, Cluj-Napoca* 280
44. *Cristian Bojinovici, București* 279
45. *Vlad Dascălu, Bacău* 274
46. *Valentin Gușă, București* 269
47. *Kovychev Ruslan Alexanderovich, Rusia* 268
48. *Andrei Paponiu, Drobeta-Turnu Severin* 263
49. *Dan Ghinea, București* 249
50. *Liviu Lalescu, Craiova* 245

MINIMIZAREA funcțiilor logice

Vasile Airinei

Algoritmul Quinne - McCluskey, despre care vom vorbi în următoarele pagini, este un algoritm performant de minimizare a funcțiilor logice. Autorul a implementat acest algoritm folosind limbajul C++. Sursele programului pot fi obținute prin e-mail de la autor, la adresa airineiv@hotmail.com.

Funcții logice, fundamente teoretice

Minimizarea funcțiilor logice a constituit mereu un subiect atractiv pentru cei care lucrează în știința calculatoarelor.

Componentele electronice ale calculatorului suportă tensiuni electrice pe două niveluri: nivelul 0 logic corespunde tensiunilor din jurul valorii de 0V și nivelul 1 logic corespunde tensiunilor din jurul valorii de 5V. În cele ce urmează nivelurile de 0 logic si 1 logic le vom numi simplu 0 și 1.

Definiție

Se numește *funcție logică* o funcție de n variabile, notată:

$$f: \{0,1\}^n \rightarrow \{0, 1\} \qquad (1)$$

Ne putem imagina o astfel de funcție ca fiind un circuit logic cu n intrări (cele n variabile) și o singură ieșire (valoarea funcției f). Deoarece circuitul are n intrări, și fiecare intrare poate avea două valori (0 sau 1) rezultă că putem avea 2^n configurații ale vectorului de intrare $[x_1, x_2, \ldots, x_n]$. Pentru fiecare configurație a vectorului de intrare vom avea o valoare a funcției f, iar această valoare va fi un element al mulțimii {0, 1}.

În expresia unei funcții logice pot apărea mai multe sume și produse de variabile. O sumă va fi numită *disjuncție* logică, iar un produs va fi numit *conjuncție* logică.

Așadar, funcția f se poate scrie ca o sumă de produse, fiecare produs corespunzând unei configurații a vectorului de intrare. Această formă de scriere a funcției f se numește forma normală disjunctivă. Fiecare produs din această sumă se numește ***minterm***.

Funcția e complet definită dacă se cunosc toate valorile sale și incomplet definită dacă unele nu sunt cunoscute.

Pentru simplitate vom trata mai întâi cazul în care funcția este complet definită. Vom numi *literal* o variabilă logică sau negația acesteia.

Apare, evident, interesul ca expresia funcției logice să fie cât mai simplă. Cu cât expresia funcției logice este mai simplă, cu atât costul componentelor hardware (porți logice, *buffer*-e) pentru implementarea funcției va fi mai scăzut.

S-au descoperit câteva metode de minimizare a funcțiilor logice (metoda *Quinne - McCluskey*, metoda hărții *Veitch - Karnaugh*). În continuare vom prezenta prima dintre aceste metode.

Metoda Quinne - McCluskey

Metoda Quinne - McCluskey este folosită pentru a reduce forma normală disjunctivă a funcției cu scopul obținerii unei sume cu număr minim de termeni. Procedura constă în doi pași principali:

1. Aplicarea sistematică a teoremei $XY + XY' = X$ (unde Y' reprezintă valoarea negată a lui Y) pentru a elimina *literalii* redundanți. Termenii rezultați sunt numiți ***implicanți primi***.
2. Se folosește o *hartă a implicanților primi* pentru a selecta o mulțime minimală de *implicanți primi* reprezentând funcția simplificată, care conține un număr minim de *literali*.

Determinarea implicanților primi

Pentru a determina expresia minimă a sumei de produse cu ajutorul metodei *Quinne - McCluskey*, funcția trebuie dată ca o sumă de *mintermi*. La primul pas al algoritmului sunt determinați toți *implicanții primi* ai funcției prin combinarea *mintermilor*. *Mintermii* sunt reprezentați în notație binară și sunt combinați folosind formula $XY + XY' = X$, unde X reprezintă un produs de *literali* și Y este un singur *literal*. Doi *mintermi* pot fi combinați numai dacă un singur literal din componența lor este diferit, restul fiind identici. Exemplele prezentate mai jos arată atât notațiile binare, cât și echivalentul algebric.

periți de unul sau mai mulți implicanți primi selectați. Termenii indiferenți nu sunt incluși în soluția finală fără să fie implicați în procesul de formare a unuia dintre implicanții primi selectați. Următorul exemplu va clarifica procedura.

Exemplu de simplificare a funcțiilor incomplet definite

$F(A, B, C, D) = \sum m(2, 3, 7, 9, 13) + \sum d(1, 10, 15)$
(mintermii care urmează după d sunt indiferenți)

Termenii indiferenți sunt tratați ca mintermi necesari când găsim implicanții primi:

1	0001 *	(1, 3)	00-1 *	(1, 3, 9, 11)	-0-1
2	0010 *	(1, 9)	-001 *	(2, 3, 10, 11)	-01-
3	0011 *	(2, 3)	001- *	(3, 7, 11, 15)	--11
9	1001 *	(2, 10)	-010 *	(9, 11, 13, 15)	-0-1
10	1010 *	(3, 7)	0-11 *		
7	0111 *	(3, 11)	-011 *		
11	1011 *	(9, 11)	10-1 *		
13	1101 *	(9, 13)	1-01 *		
15	1111 *	(10, 11)	101- *		
		(7, 15)	-111 *		
		(11, 15)	1-11 *		
		(13, 15)	11-1 *		

Tabelul 7

Coloanele indiferente sunt omise când formăm harta implicanților primi, așa cum se observă în tabelul 8:

	2	3	7	9	11	13
(1, 3, 9, 11)				×	×	
* (2, 3, 10, 13)	×	×				×
* (3, 7, 11, 15)		×	×		×	
* (9, 11, 13, 15)				×	×	×

* - indică un implicant prim esențial
$F = B'C + CD + AD$

Tabelul 8

Funcția originală a fost incomplet definită; expresia finală simplificată pentru F este definită pentru toate combinațiile de valori pentru A, B, C, D și, prin urmare, este complet definită. În procesul simplificării am luat în considerare valorile indiferente în tabela de adevăr a lui F. Dacă înlocuim fiecare term în expresia finală a lui F prin suma de mintermi corespunzătoare, rezultatul este:

$$F = (m_2 + m_3 + m_{10} + m_{11}) + (m_3 + m_7 + m_{11} + m_{15}) + (m_9 + m_{11} + m_{13} + m_{15}),$$

unde termenii duplicați sunt evidențiați.

Deoarece m_{10} și m_{15} apar în expresie, în timp ce m_1 nu apare, rezultă că termenii indiferenți în tabela de adevăr a lui F au atribuite valorile:

- pentru $ABCD = 0001$, $F = 0$;
- pentru $ABCD = 1010$, $F = 1$;
- pentru $ABCD = 1111$, $F = 1$.

Observații privitoare la codul sursă al programului care implementează algoritmul

Structurile de date necesare implementării algoritmului au fost construite pe baza bibliotecii de clase *MFC* (*Microsoft Foundation Classes*). Aplicația are facilități deosebite: calculează timpul de execuție - care pentru un număr de variabile mai mare ca 21 ia valori considerabile - și afișează memoria consumată la executarea algoritmului. Aceste facilități au fost realizate folosind funcții din programarea aplicațiilor *Windows*. Funcția logică, dată ca o listă de numere zecimale, este preluată dintr-un fișier de intrare (care are extensia .IN), iar expresia minimizată a funcției logice este afișată într-un fișier de ieșire (care are extensia .OUT).

Concluzii

Pornind de la implementarea acestui algoritm se pot realiza aplicații deosebite pentru cei care lucrează în domeniul *hardware*.

Bibliografie:

Edward J. Mc. Cluskey, *Logic design. Principles*, Prentice Hall Inc., Englewood Cliffs, N.J., 1986

Dl. Airinei Vasile este absolvent al Facultății de Automatică și Calculatoare din Iași. În prezent este inginer programator la S.C. Hidroconstrucția S.A., sucursala Sebeș. Poate fi contactat prin e-mail la airineiv@yahoo.com.

Definiții...

Hardware: componentele sistemului care pot fi lovite cu piciorul;
Software: componentele sistemului care nu funcționează;
Memorie: componenta sistemului care este întotdeauna insuficientă;
Imprimantă: componenta care se blochează doar atunci când nu este supravegheată de utilizator;
Procesor: componenta sistemului care este depășită din punct de vedere tehnologic;
Cablu: componenta sistemului care este întotdeauna prea scurtă;
BackUp: operație care nu este efectuată niciodată la timp;
Restaurare: operație care funcționează perfect până în momentul în care este nevoie de ea;

Caracterizarea sistemelor NESECVENȚIALE

Anca Vasilescu, Oana Georgescu

În episoadele anterioare am luat în discuție aspecte generale cu privire la problematica sistemelor nesecvențiale: probleme practice, clasice, specifice calculului paralel și concurent și, respectiv, clasificări ale sistemelor nesecvențiale. În acest episod vom defini conceptele specifice, insistând asupra comunicării datelor și sincronizării proceselor. Toate aceste elemente se vor constitui ca un fundament pentru abordarea în episodul următor a mecanismelor de comunicare între procese.

Un program concurent poate fi privit ca un ansamblu de componente care pot fi executate simultan. La o primă abordare, aceste componente sunt procesele care comunică între ele. În acest context sunt corecte ambele definiții ale conceptului de proces și anume: *un proces este un calcul care se poate executa paralel sau concurent cu alte calcule*, respectiv *un proces este o secvență de program în execuție*.

O condiție fundamentală a execuției concurente a programelor se poate enunța astfel: *pentru fiecare proces și pentru fiecare două instrucțiuni mașină consecutive ale procesului, intervalul de timp dintre executările lor este neprecizat, dar finit*.

Execuția concurentă a proceselor poate crea o serie de surprize celui obișnuit cu programarea secvențială. Dintre acestea, poate cea mai mare surpriză este legată de caracterul ***nedeterminist*** al execuției concurente. Comportarea nedeterministă constă în faptul că un același program concurent, pe un același set de date de intrare poate furniza la ieșire "rezultate" diferite. De cele mai multe ori, diferențele dintre ieșiri sunt cauzate de ordinea în care intră în execuție procesele active, din diferențele care pot apărea între vitezele de execuție ale diferitelor entități fizice implicate în execuția programului curent.

De asemenea, un program concurent poate fi privit ca un program care, în timpul execuției sale, creează mai multe procese care se execută într-un paralelism abstract (nu neapărat pe procesoare distincte). Existența unui singur procesor fizic impune alocarea unor ***cuante de timp*** pentru fiecare proces activ. Politica de alocare poate fi aleatoare sau poate respecta disciplina unei anumite structuri de date (coadă, coadă de priorități etc.).

Între procesele unui program concurent apar interacțiuni determinate în special de nevoia proceselor de cooperare. Aceste interacțiuni sunt de tip comunicare și/sau sincronizare. În paragrafele următoare se va arăta că aceste două concepte sunt interdependente.

În continuare vom caracteriza sistemele paralele și concurente prin definirea și descrierea conceptelor fundamentale, specifice:

- interacțiunea proceselor;
- comunicarea datelor și sincronizarea proceselor;
- proprietățile de atomicitate și granularitate, respectiv siguranță și vivacitate;
- excluderea reciprocă;
- interblocarea.

Fiecare dintre aceste caracteristici a apărut sub diferite aspecte în episoadele anterioare, începând chiar cu enunțarea problemelor clasice de concurență. Mai mult, în episodul următor, aceste proprietăți vor sta la baza evaluării instrumentelor pentru programarea nesecvențială, considerate în diferite limbaje de programare.

Interacțiunea proceselor

Interacțiunea proceselor depinde pe de o parte de aspecte legate de cooperarea și/sau competiția între procesele respective, iar pe de altă parte, depinde de deosebirea arhitecturală, structurală între sistemele cu partajare de date și cele fără partajare de date. Interacțiunea proceselor este

Exemple

AB'CD' + AB'CD = AB'C

1 0 1 0 1 0 1 1 = 1 0 1 _

(liniuța indică lipsa unei variabile).

Dacă în expresia de mai sus se înlocuiește produsul de *literali AB'C* cu *X*, și *literalul D* cu *Y* se obține formula *XY + XY' = X*.

A'BC'D + A'BCD' (nu pot fi combinați)

0 1 0 1 0 1 1 0 (nu pot fi combinați)

Pentru a găsi toți *implicanții primi*, vor fi luate în considerare toate perechile de *mintermi* și dacă este posibil, vor fi combinate. Pentru reducerea numărului de comparări, *mintermii* binari sunt sortați în grupe, în funcție de numărul biților de 1 din reprezentarea binară a fiecărui *minterm*.

În cele ce urmează vom scrie o funcție logică sub forma unei liste de numere zecimale. În listă vor apărea toți echivalenții zecimali ai acelor valori binare de la intrare pentru care funcția ia valoarea 1. Astfel, dacă avem o funcție logică de patru variabile, care pentru un vector de intrare [*a*, *b*, *c*, *d*] = 1011 scoate la ieșire valoarea 1, atunci în lista de numere zecimale care definește funcția va apărea echivalentul zecimal al numărului binar 1011, adică 11.

De exemplu, funcția logică:

$f(a, b, c, d) = \varphi m(0, 1, 2, 5, 6, 7, 8, 9, 10, 14)$ (2)

este reprezentată de următoarea listă de *mintermi*:

grupul 0: 0 0000
grupul 1: 1 0001
2 0010
8 1000
grupul 2: 5 0101
6 0110
9 1001
10 1010
grupul 3: 7 0111
14 1110

În lista de mai sus, termenii din grupa 0 nu au nici un bit cu valoarea 1, termenii din grupa 1 au câte un bit cu valoarea 1, cei din grupa 2 au câte doi biți cu valoarea 1, iar cei din grupa 3 au câte trei biți cu valoarea 1.

Așa cum am afirmat anterior, doi *mintermi* pot fi combinați dacă ei diferă în exact o variabilă.

Vom spune că două grupe sunt adiacente dacă numerele care reprezintă grupa diferă printr-o unitate. De exemplu, grupele 0 și 1 sunt adiacente; la fel, grupele 3 și 4. Grupele 0 și 2 sunt neadiacente

Compararea termenilor din grupe neadiacente este inutilă, deoarece acei termeni vor diferi întotdeauna în cel puțin două variabile și nu pot fi combinate folosind relația *XY + XY' = X*. Similar, compararea termenilor din aceeași grupă este inutilă, deoarece doi termeni cu același număr de biți având valoarea 1 vor diferi în cel puțin două variabile. Așadar, numai termenii din grupe adiacente vor fi comparați.

Întâi vom compara termenii din grupa 0 cu termenii din grupa 1. Termenii 0000 și 0001 pot fi combinați pentru a elimina variabila *d*, rezultând termenul 000-. Analog termenii 0000 și 0100 pot fi combinați și se obține termenul 00-0 (se elimină variabila *c*). Termenii rezultați sunt prezentați în coloana a doua a tabelului 1.

Numerele zecimale corespunzătoare unor termeni care pot fi combinați diferă printr-o putere a lui 2 (1, 2, 4, 8, 16 etc.). Această afirmație este adevărată deoarece, dacă reprezentările binare diferă în exact o coloană, atunci diferența numerelor corespunzătoare conține un singur bit cu valoarea 1 pe poziția în care există diferențe. Un număr binar în a cărui reprezentare binară există un singur bit cu valoarea 1 este o putere a lui 2.

Deoarece compararea membrilor grupului 0 cu membrii grupurilor 2 și 3 este inutilă, trecem la compararea termenilor din grupurile 1 și 2. Comparând *mintermul* 0001 cu toți termii din grupul 2, observăm că poate fi combinat cu termii 0101 și 1001, dar nu și cu termii 0110 și 1010. Similar, termul 0010 poate fi combinat numai cu termii 0110 și 1010. Termul 1000 poate fi combinat doar cu termii 1001 și 1010. Termii obținuți apar în coloana a doua. De fiecare dată când combinăm un term cu un altul, acesta este "bifat". Un term poate fi folosit de mai multe ori deoarece *X + X = X*. Chiar dacă doi termi au fost deja comparați cu alți termi, ei trebuie comparați și combinați între ei dacă e posibil. Această operație este necesară deoarece termul rezultat poate fi necesar pentru formarea soluției. În această fază, putem genera termi redundanți, dar aceștia vor fi eliminați ulterior.

Încheiem prima coloană prin compararea termilor din grupele 2 și 3. Noi termeni sunt formați prin compararea termilor 0101 și 0111, 0110 și 0111, sau 1010 și 1110.

	coloana 1	coloana 2	coloana 3
grupa 0	0 0000 *	0, 1 000- *	0, 1, 8, 9 -00-
		0, 2 00-0 *	0, 2, 8, 10 -0-0
grupa 1	1 0001 *	0, 8 -000 *	~~0, 8, 1, 9 -00-~~
	2 0010 *		~~0, 8, 2, 10 -0-0~~
	8 1000 *	1, 5 0-01 *	
		1, 9 -001 *	2, 6, 10, 14 --10
grupa 2	5 0101 *	2, 6 0-10 *	~~2, 10, 6, 14 --10~~
	6 0110 *	2, 10 -010 *	
	9 1001 *	8, 9 100- *	
	10 1010 *	8, 10 10-0 *	
grupa 3	7 0111 *	5, 7 01-1	
	14 1110 *	5, 7 011- *	
		6, 14 -110 *	
		10, 14 -110 *	

Tabelul 1: Determinarea implicanților primi

Observăm că termenii din coloana a doua sunt împărțiți în grupe în funcție de numărul de biți cu valoarea 1 din reprezentarea binară a fiecărui term. Aplicăm formula *XY + XY' = X* pentru a combina perechi de termeni în coloana a doua. Pentru a combina doi termi ei trebuie să aibă aceleași variabile termenii trebuie să difere în exact una dintre

aceste variabile. Așadar, este necesară doar compararea termenilor care au liniuța pe aceeași poziție și al căror număr de biți cu valoarea 1 diferă exact cu 1. *Termii* aflați în prima grupă în a doua coloană vor fi comparați numai cu *termii* din grupa a doua care au liniuța pe aceeași poziție. *Termul* 000- (0,1) poate fi combinat numai cu *termul* 100- (8, 9) și se obține -00-. Acesta e echivalentul algebric al relației $a'b'c' + ab'c' = b'c'$. *Termul* rezultat este listat în coloana a treia prin descrierea 0, 1, 8, 9. *Termul* (0, 2) se poate combina numai cu (8, 10) și *termul* (0, 8) se combină numai cu (1, 9) și (2, 10). Și în acest caz, *termii* care au fost deja combinați sunt bifați. Comparând *termii* din a doua și a treia grupă în a doua coloană observăm că (2, 6) poate fi combinat cu (10, 14), iar (2, 10) poate fi combinat cu (6, 14).

Observăm că sunt trei perechi de *termi* duplicate în coloana a treia. Acești *termi* duplicați au fost formați în fiecare caz prin combinarea aceluiași set de patru *mintermi* în ordine diferită. După eliminarea *termilor* duplicați, comparăm *termii* din cele două grupe din a treia coloană. Întrucât nici o combinație nu mai este posibilă, procesul se termină. În general, vom păstra *termii* comparați și vom forma noi grupe de *termi* și noi coloane până când nu se mai pot combina *termi*.

Termii care nu mai pot fi combinați se numesc *implicanți primi*. Deoarece fiecare *minterm* este inclus în cel puțin unul din *implicanții primi*, funcția este egală cu suma *implicanților primi*. În acest exemplu, avem:

$$f = a'c'd + a'bd + a'bc + b'c' + b'd' + cd' \quad (3)$$

(1,5) (5,7) (6,7) (0,1,8,9) (0,2,8,10) (2,6,10,14)

În expresia de mai sus, fiecare *term* are număr minim de *literali*, dar numărul de *termi* nu este minim. Folosind teorema pentru eliminarea *termilor* redundanți obținem:

$$f = a'bd + b'c' + cd' \quad (4),$$

care este expresia minimă a lui *f* scrisă ca sumă de produse. În secțiunea următoare vom prezenta o metodă mai performantă pentru eliminarea *implicanților primi* redundanți, folosind o hartă a implicanților primi.

În continuare, vom defini noțiunile de ***implicant*** și ***implicant prim***.

Definiție:

Pentru o funcție *F* de *n* variabile, un produs de *termi P* este un ***implicant*** al lui *F* dacă pentru fiecare combinație de valori de *n* variabile pentru care $P = 1$, *F* are, de asemenea, valoarea 1.

Cu alte cuvinte, dacă există anumite combinații de valori pentru variabile, $P = 1$ și $F = 0$, atunci *P* nu este ***implicant*** al lui *F*. De exemplu, considerăm funcția :

$$F(a, b, c) = a'b'c' + ab'c' + ab'c + abc = b'c' + ac \quad (5)$$

Dacă $a'b'c' = 1$, atunci $F = 1$, dacă $ac = 1$, atunci $F = 1$ etc. Astfel, *termii* $a'b'c'$, ac, sunt *implicanți* ai lui *F*. În acest exemplu, bc nu este un *implicant* al lui F deoarece, dacă $a = 0$ și $b = c = 1$, atunci $bc = 1$ și $F = 0$. În general, dacă *F* este scrisă sub forma sumei de produse, fiecare *term* este un *implicant*. Fiecare *minterm* al lui *F* este, de asemenea, un *implicant* al lui *F* și, prin urmare, orice *term* format prin combinarea a doi sau mai mulți *mintermi* va fi un *implicant* al lui *F*. De exemplu, în tabelul 1, toți *termii* sunt listați în fiecare coloană în care sunt implicanți ai funcției dată de ecuația (2).

Definiție

Un ***implicant prim*** al funcției *F* este un *implicant* din care, dacă prin eliminarea unui *literal* din componența acestuia nu se obține alt *implicant*.

În ecuația (5), *implicantul* $a'b'c'$ nu este *implicant prim* deoarece a' poate fi eliminat și *termenul* rezultat ($b'c'$) este un *implicant* al lui *F*. *Implicanții* $b'c'$ și ac sunt *implicanți primi* deoarece, dacă ștergem un *literal* din fiecare *term*, *termul* rezultat nu va mai fi *implicant* al lui *F*. Fiecare *implicant prim* al funcției are un număr minim de *literali* în sensul că nu mai pot fi eliminați *literali* prin combinarea cu alți *termi*.

Expresia minimă sumă de produse pentru o funcție constă dintr-o sumă de anumiți implicanți primi ai funcției. Dacă o expresie sumă de produse conține un term care nu e implicant prim, atunci acea expresie nu este minimizată, deoarece acel term poate fi înlocuit cu un implicant prim. Această afirmație este adevărată deoarece termii neprimi nu conțin un număr minim de *literali* - ei pot fi combinați cu *mintermi* adiționali pentru a forma un implicant prim care are mai puțini *literali* decât *termii* neprimi. Un term neprim într-o sumă de produse poate fi deci înlocuit cu un implicant prim, care reduce numărul de *literali* și simplifică expresia.

Harta implicanților primi

A doua parte a procedurii *Quinne - McCluskey* dezvoltă o hartă pentru a selecta setul minim de implicanți primi. Un implicant prim este egal cu o sumă de mintermi și se spune că implicantul prim acoperă acei mintermi. Dacă un implicant prim acoperă un minterm dat, un × este plasat la intersecția liniei și coloanei corespunzătoare. Tabelul 2 prezintă harta implicanților primi dedusă din tabelul 1. Toți implicanții primi (termi care nu au fost eliminați din tabelul 1) sunt listați în partea stângă.

i.p. \ m	0	1	2	5	6	7	8	9	10	14
(0, 1, 8, 9)	×	×					×	⊗		
(0, 2, 8, 10)	×		×				×		×	
(2, 6, 10, 14)			×		×				×	⊗
(1, 5)		×		×						
(5, 7)				×		×				
(6, 7)					×	×				

m - mintermi

i.p. - implicanți primi

Tabelul 2: Harta implicanților primi

În prima linie, ×-urile sunt plasate în coloanele 0, 1, și 9 deoarece implicantul $b'c'$ a fost format din suma d

mintermi 0, 1, 8, 9. Similar, ×-urile sunt plasate în coloanele 0, 2, 8, 10 corespunzător implicantului prim *b'd'* ș.a.m.d.

Dacă un *minterm* este acoperit numai de un implicant prim, atunci cel din urmă este numit *implicant prim esențial* și trebuie inclus în suma minimă de produse. Implicanții primi esențiali sunt ușor de găsit folosind harta implicanților primi. Dacă o coloană dată conține numai un singur ×, atunci linia pe care se află ×-ul corespunde unui implicant prim esențial. În tabelul 2, coloanele 9 și 14 conțin fiecare câte un ×, așadar implicanții primi *b'c'* și *cd'* sunt esențiali.

De fiecare dată când un implicant prim este selectat pentru includerea în suma minimă, linia corespunzătoare este eliminată. După realizarea acestei operații, coloanele care corespund tuturor *mintermilor* acoperiți de acest implicant prim vor fi eliminate.

Tabelul 3 ilustrează harta rezultată când sunt eliminați implicanții primi esențiali, împreună cu liniile și coloanele corespunzătoare din tabelul 2. În continuare, trebuie ales un set minim de implicanți primi pentru a acoperi coloanele rămase. În acest exemplu, *a'bd* acoperă cele două coloane rămase, deci poate fi implicantul prim ales. Suma minimă de produse care rezultă este:

$$f = b'c' + cd' + a'bd,$$

care este identică cu cea din ecuația (4).

Observăm că, chiar dacă termul *a'bd* este inclus în suma minimă de produse, *a'bd* nu este un implicant prim esențial. El reprezintă suma de *mintermi* m_5, m_7; m_5 este acoperit și de *a'c'd*, iar m_7 este acoperit și de *a'bc*.

i.p. \ m	0	1	2	5	6	7	8	9	10	14
(0, 1, 8, 9)	×						⊗			
(0, 2, 8, 10)	×		×				⊗		×	
(2, 6, 10, 14)			×		×				×	⊗
(1, 5)		×		×						
(5, 7)				×		×				
(6, 7)					×	×				

Tabelul 3

După construirea hărții implicanților primi, anumiți *mintermi* pot fi acoperiți de numai un singur implicant prim, în timp ce alți *mintermi* pot fi acoperiți de doi sau mai mulți implicanți primi. Un implicant prim este esențial (sau necesar) dacă el conține un *minterm* care nu e acoperit de alt implicant al lui *f*. Implicanții primi esențiali sunt aleși la început deoarece toți implicanții primi esențiali trebuie incluși în fiecare sumă minimă. După ce au fost aleși implicanții primi esențiali, *mintermii* pe care îi acoperă pot fi eliminați din harta implicanților primi traversând coloanele corespunzătoare. Dacă implicanții primi esențiali nu acoperă toți *mintermii*, atunci sunt necesari implicanți primi neesențiali adiționali. În cazuri simple, implicanții primi neesențiali necesari pentru formarea soluției minime pot fi selectați prin probe. Pentru hărți mai mari, trebuie folosite proceduri pentru reducerea hărții care vor fi prezentate ulterior. Anumite funcții au două sau mai multe expresii minime sume de produse, fiecare având același număr de *termi* și *literali*. Următorul exemplu ilustrează o asemenea funcție.

Exemplu cu hartă ciclică de implicanți primi

O hartă a implicanților care are două sau mai multe ×-uri în fiecare coloană este numită ***hartă ciclică***. Următoarea funcție are o asemenea hartă:

$$f = \varphi m(0, 1, 2, 5, 6, 7) \qquad (6)$$

Formarea implicanților primi se face astfel:

0 000 *	0,1	00-
1 001 *	0,2	0-0
2 010 *	1,5	-01
5 101 *	2,6	-10
6 110 *	5,7	1-1
7 111 *	6,7	11-

Tabelul 4

Tabelul 5 ilustrează harta implicanților primi care rezultă.

	0	1	2	5	6	7
① → (0, 1)	×	×				
(0, 2)	×		×			
(1, 5)		×		×		
② → (2, 6)			×		×	
③ → (5, 7)				×		×
(6, 7)					×	×

Tabelul 5

Toate coloanele au două ×-uri, așadar trebuie efectuate probe. (0, 1) și (0, 2) acoperă coloana 0; vom încerca (0, 1). După traversarea liniei (0, 1) și coloanele 0 și 1, vom examina coloana 2 care e acoperită de (0, 2) și (2, 6). Cea mai bună alegere este (2, 6) deoarece acoperă două din coloanele rămase, în timp ce (0, 2) acoperă numai una din coloanele rămase. După traversarea liniei (2, 6) și coloanele 2 și 6, observăm că (5, 7) acoperă coloanele rămase și completează soluția. Așadar o soluție este:

$$F = a'b' + bc' + ac.$$

Totuși, nu avem garanția că această soluție este minimă. Trebuie să reluăm algoritmul, folosind un alt implicant care acoperă coloana 0. Rezultă tabelul 6.

		0	1	2	5	6	7
P1	(0, 1)	×	×				
P2	(0, 2)	×		×			
P3	(1, 5)		×		×		
P4	(2, 6)			×		×	
P5	(5, 7)				×		×
P6	(6, 7)					×	×

Tabelul 6

În sfârșit, obținem soluția:

$$F = a'c' + b'c + ab$$

Întrucât această soluție are același număr de termeni și același număr de literali ca expresia lui F derivată din tabelul 5, avem 2 soluții sume de produs minime pentru această problemă.

Metoda Petrick

Aceasta este o metodă pentru determinarea tuturor soluțiilor sumă de produs din harta implicanților primi. Exemplul corespunzător tabelelor 5 și 6 are două soluții minime. Creșterea numărului de variabile duce la creșterea semnificativă a numărului de implicanți primi și a complexității hărții implicanților primi. În asemenea cazuri, pentru obținerea soluției minime este necesară efectuarea unui număr mare de probe.

Metoda Petrick este un mod mai sistematic de găsire a tuturor soluțiilor minime din harta implicanților primi. Înaintea aplicării *metodei Petrick*, toți implicanții primi esențiali și mintermii care îi acoperă trebuie șterși de pe hartă.

Vom ilustra *metoda Petrick* folosind tabelul 6.

Întâi vom eticheta liniile P_1, P_2, P_3 etc. Vom construi o funcție logică P, care e adevărată când toți mintermii în hartă au fost acoperiți. Fie P_1 o variabilă logică care este adevărată când implicantul prim din linia P_1 este inclus în soluție; fie P_2 o variabilă logică care e adevărată când implicantul prim din linia P_2 este inclus în soluție etc. Deoarece coloana 0 conține ×-uri în liniile P_1 și P_2, trebuie să alegem una din liniile P_1 sau P_2 pentru a acoperi mintermul 0. Expresia $(P_1 + P_2)$ trebuie să fie adevărată. Pentru a acoperi mintermul 1, trebuie să alegem liniile P_1 sau P_3; atunci $(P_1 + P_3)$ trebuie să fie adevărată. Pentru a acoperi mintermul 2, $(P_2 + P_4)$ trebuie să fie adevărată. Similar, pentru a acoperi mintermii 5, 6 și 7, expresiile $(P_3 + P_5)$, $(P_4 + P_6)$ și $(P_5 + P_6)$ trebuie să fie adevărate. Obținem expresia:

$$P = (P_1 + P_2) \cdot (P_1 + P_3) \cdot (P_2 + P_4) \cdot (P_3 + P_5) \cdot (P_4 + P_6) \cdot (P_5 + P_6) = 1.$$

Această expresie indică faptul că trebuie să alegem linia P_1 sau P_2 și linia P_1 sau P_3 și linia P_2 sau P_4 etc.

Următorul pas este de a reduce P la o sumă minimă de produse. Operația este mai ușor de realizat deoarece nu există complemente. Folosim relația $(X + Y) \cdot (X + Z) = X+YZ$ și legea distributivității:

$$\begin{aligned} P &= (P_1 + P_2P_3) \cdot (P_4 + P_2P_6) \cdot (P_5 + P_3P_6) = \\ &= (P_1P_4 + P_1P_2P_6 + P_2P_3P_4 + P_2P_3P_6) \cdot (P_5 + P_3P_6) = \\ &= P_1P_4P_5 + P_1P_2P_5P_6 + P_2P_3P_4P_5 + P_2P_3P_5P_6 + \\ &\quad + P_1P_3P_4P_6 + P_1P_2P_3P_6 + P_2P_3P_4P_6 + P_2P_3P_6. \end{aligned}$$

În continuare, vom folosi relația $X + XY = X$ pentru a elimina termenii redundanți din P și obținem:

$$P = P_1P_4P_5 + P_1P_2P_5P_6 + P_2P_3P_4P_5 + P_1P_3P_4P_6 + P_2P_3P_6.$$

Pentru a acoperi toți mintermii trebuie să fie satisfăcută relația $P = 1$. Așadar, vom alege fie liniile P_1, P_4 și P_5, fie liniile P_1, P_2, P_5 și P_6, fie liniile P_2, P_3, P_4 și P_5, fie liniile P_1, P_3, P_4 și P_6, fie liniile P_2, P_3 și P_6. Sunt cinci soluții posibile, iar numai două dintre ele au un număr minim de linii. În concluzie, cele două soluții cu număr minim de implicanți primi sunt: liniile P_1, P_4 și P_5 sau liniile P_2, P_3 și P_6. Prima alegere conduce la relația:

$$F = a'b' + bc' + ac.$$

Alegând a doua opțiune, obținem:

$$F = a'c' + b'c' + ab.$$

Pe scurt, *metoda Petrick* poate fi descrisă astfel:

- Se reduce harta implicanților primi prin eliminarea liniilor implicanților primi esențiali și coloanele corespunzătoare;
- Se etichetează liniile hărții reduse de implicanții primi P_1, P_2, P_3 etc;
- Se determină o funcție logică P care este adevărată când toate coloanele sunt acoperite. P este, de fapt, un produs de sume de termi, fiecare sumă având forma $(P_{i_0} + P_{i_1} + ...)$, unde P_{i_0}, P_{i_1}, ... reprezintă liniile care acoperă coloana i;
- Se reduce P la o sumă minimă de produse prin înmulțire și aplicarea legii $X + XY = X$;
- Fiecare term din rezultat reprezintă o soluție, care este un set de linii care acoperă toți mintermii în tabel. Pentru a determina soluțiile minime, găsim acei termi care conțin un număr minim de variabile. Fiecare din acești termi reprezintă o soluție cu un număr minim de implicanți primi;
- Pentru fiecare din termii găsiți la pasul 5, numărăm literalii din fiecare implicant prim și reținem numărul de literali. Alegem termul sau termii care corespund unui număr minim de literali, și avem astfel sumele corespunzătoare de implicanți primi.

Aplicarea *metodei Petrick* este foarte utilă pentru hărți "incomode" și e relativ ușor de implementat.

Simplificarea funcțiilor incomplet definite

Termenii indiferenți corespund acelor vectori de intrare pentru care nu se cunoaște valoarea funcției logice de la ieșirea circuitului. Lista termenilor indiferenți va fi separată de lista mintermilor în expresia funcției logice. De exemplu, dacă avem o funcție logică de patru variabile a cărei valoare nu se cunoaște pentru vectorul de intrare [a, b, c, d] = 1000, atunci echivalentul zecimal al valorii binare 1000 adică 8, va face parte din lista termenilor indiferenți.

În cazul în care avem o funcție incomplet definită, trebuie să luăm în considerare și valorile termenilor indiferenți pentru a obține forma minimă a funcției. În această secțiune vom arăta cum se modifică procedura *Quinne McCluskey* pentru a obține soluția minimă când sunt prezenți termeni indiferenți. În scopul găsirii implicanților primi, vom trata termenii indiferenți ca și cum ar fi mintermi. Astfel, ei pot fi combinați cu alți mintermi pentru a elimina cât mai mulți literali. Dacă se formează implicanți primi din termeni indiferenți, acei implicanți nu vor fi luați în considerare. În partea de sus a hărții implicanților primi nu vom trece termenii indiferenți. Când harta implicanților primi este rezolvată toți mintermii necesari vor fi ac

subiect deosebit de important în tematica sistemelor concurente și, de aceea, el se va regăsi atât în această secțiune, cât și în tratarea comunicării datelor și sincronizării proceselor concurente.

Concret, la nivelul sistemului nesecvențial care se analizează, interacțiunea proceselor depinde în mare măsură de localizarea proceselor în funcție de structura modulară a *soft*-ului sistemului respectiv.

Putem prezenta această structură *software* ținând cont de proprietățile zonelor de memorie în care se execută procesele curente. Din acest punct de vedere se deosebesc următoarele situații:

- procesele pot partaja spațiu de adresă;
- procesele se pot executa în spații de adresă separate, distincte și complet disjuncte de spațiile de adresă ale altor procese.

Dacă procesele pot partaja spațiu de adresă, atunci avem mai multe variante de organizare a execuției proceselor, în funcție de necesitatea cooperării între procesele curente și de interacțiunea între procesele care pot partaja simultan, prin acces direct, un spațiu de adresă. Astfel, se deosebesc două variante principale:

- toate procesele partajează un același spațiu de adresă și de aici rezultă că se vor executa cu partajare de date;
- procesele se execută în spații de adresă diferite (fiecare proces poate avea spațiu de adresă propriu), dar este permisă partajarea unora dintre segmentele codurilor proceselor (segmentul de cod și/sau segmentul de date); dacă segmentul de date este unul care nu poate fi partajat, atunci procesele cooperante urmează să stabilească alt protocol de comunicare, nu prin partajarea datelor; după localizarea fizică a spațiilor de adresă proprii proceselor, se pot deosebi procesele care se execută pe aceeași mașină de cele care se execută pe mașini distincte.

Un exemplu de sistem care implementează aceste variante de gestiune a proceselor este un sistem *multiuser* în care se impun cerințe de protecție a memoriei când unul dintre utilizatori are restricții de scriere sau de citire la nivelul zonelor de memorie ale altui utilizator. De asemenea, într-un sistem *multiuser* trebuie impuse restricții legate de protecția asupra segmentelor de cod comun ale componentei *software* a sistemului.

Un alt exemplu îl constituie procesele distribuite care partajează date sau cod cu alte procese din sistem. Acestea sunt procese care ocupă spații de adresă distincte, chiar situate pe mașini diferite. Orice comunicare între procesele distribuite folosește mijloacele de comunicare din rețeaua mașinilor din sistem, prin intermediul interfețelor *hardware* și pe baza protocoalelor de comunicare recunoscute în sistemul distribuit respectiv.

Dacă procesele se pot executa în spații de adresă separate, distincte și complet disjuncte de spațiile de adresă ale altor procese atunci putem deosebi două variante specifice și anume:

- toate procesele se execută în spații disjuncte;
- sunt procese care se execută în spații disjuncte.

Primul caz reprezintă o contradicție cu definiția concurenței, deci nu poate fi luat în considerare dacă se dorește implementarea unui sistem concurent.

În cel de-al doilea caz, un proces care se execută într-un spațiu de adresă disjunct de cel al oricărui alt proces este un proces care nu va partaja nici segmente de date, nici segmente de cod cu alte procese din sistem. Un astfel de proces poate doar să intermedieze execuția unei instrucțiuni prin scrierea sau citirea unui operand din propriul spațiu de adresă. Acesta este cazul proceselor distribuite care se execută pe mașini diferite într-un sistem distribuit.

Pentru detalierea acestui subiect vă propunem să consultați [1].

Toate aceste variante de comunicare între procese fac obiectul mecanismelor *IPC* (*Inter Process Communication*), pe care le vom aborda în episodul următor.

Cerințe pentru interacțiunea proceselor

Cerințele pentru interacțiunea proceselor rezultă, pe de o parte, din necesitatea ca procesele să interacționeze între ele și, pe de altă parte, din felul în care aceste interacțiuni pot fi suportate de sistemul gazdă.

Procesele pot coopera pentru îndeplinirea unei sarcini comune. Concret, un proces se poate afla în poziția de a cere altui proces furnizarea unui serviciu și eventual, ar putea să aștepte să primească acel serviciu. Exemple pentru această situație ar putea fi:

- aspectele cooperative care reies din diferite variante de implementare a localizării unui proces asociat unui anumit modul furnizor de servicii;
- cazul în care procesul de gestionare a discurilor trebuie să se sincronizeze cu procesele discurilor respective;
- prelucrarea proceselor în *pipeline* în care fiecare proces de fază trebuie să aștepte până când procesul fazei anterioare își încheie execuția.

Cooperarea între procese presupune sincronizarea lor, fapt care reiese din aspectele enumerate mai sus. Cu alte cuvinte, un proces poate fi nevoit să execute o operație de tipul `WAIT(PROCES)` pentru a se sincroniza cu alte procese. Sincronizarea între procese poate fi privită ca o extindere a sincronizării proceselor cu funcționarea componentelor *hardware* ale sistemului.

Comunicarea de date între procesele cooperante se realizează, de obicei, prin operații `SIGNAL(PROCES)`, deci prin semnale sau întreruperi.

Procesele pot concura pentru acces exclusiv la servicii sau la resurse. Din acest punct de vedere se disting două aspecte diferite și anume:

- mai mulți clienți adresează simultan cereri pentru anumite servicii;
- mai multe procese încearcă să acceseze simultan o resursă.

Implementarea sistemului concurent trebuie să prevadă și să gestioneze astfel de situații conflictuale prin stabilirea unei ordini predefinite, prin întârzierea execuției unora dintre procese, prin stabilirea unor priorități de acces la

resursele care pot deservi numai serial cererile pe care le primesc.

Procesele concurente pot să fie nevoite să execute operații **WAIT**(PROCES) înainte de a accesa o resursă partajată și trebuie să poată transmite semnale prin operații **SIGNAL**(PROCES) pentru a indica momentul în care au părăsit resursa partajată în sistem.

Procesele care se execută în spații de adrese distincte vor fi cunoscute sistemului de operare. În majoritatea cazurilor, codul acestor procese provine dintr-o sursă scrisă într-un limbaj de programare secvențial. În acest caz, operațiile **WAIT** și **SIGNAL** sunt recunoscute ca fiind *apeluri sistem*.

Procesele care partajează spații de adrese de memorie devin părți ale aceluiași program concurent și pot să fie sau nu cunoscute la nivelul sistemului de operare.

Dacă procesele nu sunt cunoscute de sistemul de operare atunci operațiile **WAIT**(PROCES) și **SIGNAL**(PROCES) vor putea fi lansate numai de la nivelul limbajului de programare.

Dacă procesele cooperează, atunci se pot dezvolta diferite mecanisme bazate pe operațiile **WAIT**(PROCES) și **SIGNAL**(PROCES) care să asigure comunicarea și sincronizarea prin construcții specifice pentru dezvoltarea aplicațiilor concurente în limbaje de programare de nivel înalt.

În cele ce urmează vom arăta că aceste două operații elementare (**WAIT**(PROCES) și **SIGNAL**(PROCES)) nu sunt suficiente pentru descrierea completă a unei scheme reale de comunicare între procese de tip *IPC*.

Comunicarea datelor și sincromizarea proceselor

Fiecare limbaj paralel trebuie să recunoască elemente specifice paralelismului fie explicit, fie implicit [2]. Practic, sunt mai accesibile limbajele de programare care nu cer construcții de limbaj pentru paralelism, dar permit procesarea paralelă printr-un paralelism implicit.

Aceasta este mai ușor de realizat în limbajele declarative, adică cele logice (*Prolog*) sau funcționale (*Lisp*), deoarece informația procedurală minimală trebuie paralelizată de un compilator "inteligent" care funcționează pe baza unor reguli de deducere și are o interacțiune foarte limitată cu programatorul.

Reprezentarea declarativă a cunoștințelor sau a problemei de rezolvat poate specifica fără echivoc o soluție. Totuși, poate fi destul de dificil să se convertească aceste cunoștințe într-un program paralel imperativ [3].

Indiferent dacă paralelismul este implicit sau explicit, trebuie să existe o cale de a crea procese paralele și o cale de a coordona activitatea acestor procese.

Uneori, procesele lucrează asupra datelor proprii și nu interacționează. Dar, atunci când procesele interschimbă rezultate, ele trebuie să comunice și să se sincronizeze între ele.

Comunicarea și sincronizarea pot fi realizate prin variabile partajate sau prin transfer de mesaje.

Comunicarea datelor în sisteme concurente

Comunicarea constă în schimbul de date (informații) între procesele care cooperează. Comunicarea poate fi identificată cu transmiterea de mesaje între procesele concurente.

Se pot identifica trei criterii de clasificare a tipurilor de comunicare:
- mijlocul de comunicare;
- gradul de sincronizare;
- modul de precizare a sursei și destinației comunicării.

În continuare vom detalia aceste clasificări. În funcție de mijlocul de comunicare avem:
- comunicare prin memorie comună;
- comunicare prin mecanisme specifice de tip monitor, semafor, resursă, canal.

Datorită faptului că o zonă de memorie utilizată în comun este o resursă *critică*, rezultă că, în acest caz, se impune programarea de secțiuni critice și implicit rezolvarea problemei excluderii mutuale.

În funcție de gradul de sincronizare avem cele două clase naturale de comunicare și anume:
- comunicarea fără sincronizare, care poate fi identificată cu transmiterea asincronă de mesaje, deci comunicare fără semnal de recepție;
- comunicarea cu sincronizare, care poate fi identificată cu transmiterea sincronă de mesaje; la rândul ei, comunicarea cu sincronizare poate fi realizată prin:
  - ♦ sincronizare simplă;
  - ♦ invocare la distanță.

În categoria *comunicare prin sincronizare simplă* putem încadra emiterea mesajului cu semnal de recepție, comunicarea prin canale sau cu punct de întâlnire de tip *rendez-vous* (vezi episodul următor). În cazul invocării la distanță ***recepția*** mesajului înseamnă atât *semnal de recepție*, cât și un răspuns de la destinatar la expeditor. Chiar și în sintaxa unor limbaje de programare concurente, invocarea la distanță apare ca și generalizare a comunicării prin canale sau ca un *rendez-vous* extins.

Pe de altă parte, invocarea la distanță reprezintă mecanismul *RPC*, (*Remote Procedure Call*) de implementare a modelului *client/server* de interacțiune între procese: mai multe procese (active) acționează în calitate de clienți asupra unui proces server (pasiv) ce urmărește în principal satisfacerea cererilor clienților (vezi episodul următor).

În funcție de modul de precizare a sursei și destinației, comunicarea se poate face prin precizare:
- explicită;
- indirectă a partenerilor implicați în comunicare.

Prin precizare indirectă se înțelege folosirea unui intermediar între procesele care comunică. De exemplu, un astfel de intermediar ar putea fi un canal de comunicare care preia mesajul de la procesul sursă, în timp ce procesul destinație așteaptă să preia mesajul de pe acel canal.

Sincronizarea proceselor în sisteme concurente

Sincronizarea este o restricție impusă în evoluția în timp a proceselor. Ea constă în planificarea execuției a două pro-

cese, în cazul în care continuarea unuia depinde de execuția celuilalt.

Pe de altă parte, conform [4], sincronizarea proceselor constă în serializarea accesului proceselor concurente la resursele partajate.

Reluând o idee din descrierea comunicării, putem clasifica sincronizarea astfel:
- cu transmitere de mesaje (comunicare);
- fără transmitere de mesaje (fără comunicare).

În a doua situație sincronizarea se realizează la nivelul zonelor de memorie comune proceselor concurente.

Modalitatea naturală pentru rezolvarea sincronizării se dovedește a fi cooperarea logică prin diverse structuri de date. Limbajele de programare oferă programatorului posibilitatea de a opta între utilizarea diverselor tipuri abstracte de date pentru sincronizare, cu sau fără comunicare.

Din punctul de vedere al nivelului de sincronizare, programatorul poate opta pentru:
- sincronizare pe generație;
- sincronizare pe condiție.

Sincronizarea pe generație constă în întârzierea unui proces până când toate procesele din aceeași generație cu el și-au încheiat activitatea. Acest model se regăsește în rezolvarea problemei *jocul vieții* sau în soluțiile paralele/concurente pentru calculul unor valori numerice date de expresii sumă.

Sincronizarea pe condiție constă în întârzierea unui proces până când este îndeplinită o anumită condiție. Această variantă este mult mai des folosită în aplicațiile practice și datorită faptului că mecanismele de implementare a sincronizării pe condiție sunt mai bine reprezentate în majoritatea limbajelor de programare cu facilități pentru concurență.

Quinn [2] afirmă că există două metode de sincronizare:
- sincronizarea de precedență;
- sincronizarea pentru excludere mutuală.

Sincronizarea de precedență asigură faptul că un eveniment nu începe până când un altul nu s-a încheiat. *Sincronizarea pentru excludere mutuală* asigură faptul că, la un moment dat, numai un proces intră în secțiunea critică a codului cu care se prelucrează datele partajate.

Proprietăți specifice

Atomicitate și granularitate

Abordarea sistemelor concurente ridică o serie de probleme care pot fi rezolvate numai apelând la entități specifice. Din acest punct de vedere trebuie spus că tratarea concurenței trebuie să facă distincție între acțiuni concurente singulare (*single concurrent actions*) și acțiuni concurente compuse (*concurrent composite actions*). Putem defini o acțiune singulară ca fiind o operație de citire/scriere din/în memoria principală, dar problematica sistemelor concurente nu se poate folosi numai de acest tip de operații. Un motiv ar fi simplul fapt că procesele concurente sunt corect și complet descrise prin astfel de operații singulare numai dacă se poate presupune că orice operație este garantată de resursele *hardware* că este indivizibilă, adică atomică. Această condiție este prea restrictivă în condițiile unui sistem concurent general. De aceea, definiția unei acțiuni concurente singulare trebuie extinsă la operații de nivel mai înalt decât citirea/scrierea din/în memorie.

Potrivit lui ***Andrews*** [5], o acțiune atomică este o secvență de una sau mai multe stări care apare ca fiind indivizibilă, în sensul că nici un proces extern ei nu are acces la nici una dintre stările intermediare. Detaliind, *Andrews* definește două tipuri de acțiuni atomice:
- o acțiune atomică este de granularitate fină dacă poate fi implementată direct printr-o instrucțiune mașină unică;
- o acțiune atomică poate fi o secvență de acțiuni atomice de granularitate fină, indivizibilă.

O acțiune concurentă compusă rezultă prin combinarea acțiunilor singulare într-o operație de nivel înalt. Considerentele legate de aceste aspecte ale concurenței deschid două direcții distincte și anume:
- legarea acestor noțiuni de structura modulară a componentei *software* a unui sistem, în particular de tipurile abstracte de date recunoscute la nivelul sistemului respectiv;
- stabilirea legăturilor între acțiunile singulare astfel încât să se poată decide dacă ele pot forma sau nu acțiuni compuse.

În [5] se afirmă că o acțiune atomică realizează o transformare indivizibilă de stare. Aceasta înseamnă că orice stare intermediară care ar putea să existe în implementarea acțiunii nu trebuie să fie vizibilă pentru alte procese.

Proprietatea de granularitate reprezintă numărul de procesoare din sistem sau, dacă se referă la o aplicație, atunci granularitatea reprezintă numărul de procese care compun aplicația respectivă.

O acțiune atomică ***fine-grained*** este implementată direct de componenta *hardware* a sistemului pe care se execută programul concurent.

Uneori este nevoie să folosim un mecanism de sincronizare pentru a construi o acțiune atomică ***coarse-grained***, care este o secvență de acțiuni atomice *fine-grained* care apare ca fiind indivizibilă.

Exemple

Presupunem că o bază de date conține două valori x și y și că aceste valori trebuie să fie în fiecare moment egale. Cu alte cuvinte, nici un proces nu are acces la baza de date dacă $x \neq y$. Pentru a asigura această restricție, orice proces care modifică valoarea lui x trebuie să modifice și valoarea lui y, ca parte a aceleiași acțiuni atomice.

Fie un proces care inserează elemente într-o coadă reprezentată printr-o listă de legături. Un alt proces extrage elemente din listă, presupunând că există elemente în listă. Două variabile referă capul și, respectiv, coada listei. Valorile acestor variabile sunt modificate în timpul executării

operației de inserare, respectiv de extragere a elementelor din listă. Dacă lista conține un singur element, atunci o inserare și o extragere simultane pot crea conflicte, lista ajungând într-o stare instabilă. De aceea, inserarea și, respectiv, extragerea elementelor din listă trebuie să fie acțiuni atomice. Mai mult, dacă lista este vidă atunci trebuie să se întârzie executarea unei extrageri până când va fi executată o inserare.

Siguranță și vivacitate

Un atribut adevărat pentru oricare execuție a unui program este o proprietate a programului respectiv. Orice proprietate se poate exprima pe baza a două proprietăți speciale și anume:

- siguranța (*safety*);
- vivacitatea, viabilitatea (*liveness*) [5], [6], [1].

În general, un program este sigur dacă "execuția sa nu implică apariția unor evenimente neplăcute" și este viabil dacă "este de așteptat să apară un rezultat bun".

Un program secvențial este sigur dacă starea finală este corectă și este viabil dacă execuția lui se termină.

Caracterul nedeterminist al programelor concurente nu ne permite să extindem cu ușurință aceste fomulări de la programele secvențiale la programele concurente. De aceea, pentru un program concurent, definițiile proprietăților de siguranță și viabilitate vor folosi conceptele specifice aplicațiilor concurente.

Astfel, siguranța unui program concurent constă în asigurarea excluderii mutuale și absența interblocării (interblocarea apare când procesele așteaptă îndeplinirea unor condiții care nu pot fi satisfăcute), iar viabilitatea se exprimă prin faptul că un proces în așteptare va fi semnalizat în curând și astfel va putea să-și continue execuția.

Viabilitatea unui program concurent depinde de:

- corectitudinea alocării resurselor (*fairness*) - un proces va obține șansa de a intra în execuție, independent de starea altor procese;
- așteptarea activă (*starvation*) - un proces activ, care este blocat pe un timp nedefinit de unitatea de planificare (deși este teoretic capabil să se execute), va fi ales la un moment dat pentru a-și continua execuția. Această condiție este una esențială și pentru evitarea stării de blocare globală a sistemului (*deadlock*).

Excluderea reciprocă

Din considerente practice s-a ajuns la necesitatea ca, în contexte concrete, o secvență de instrucțiuni din programul de cod al unui proces să poată fi considerată ca fiind indivizibilă. Prin proprietatea de a fi indivizibilă înțelegem că după începerea execuției instrucțiunilor din această secvență, nu se va mai executa nici o acțiune a unui alt proces, până la terminarea execuției instrucțiunilor din acea secvență.

Trebuie să subliniem un aspect fundamental și anume că această secvență de instrucțiuni este o entitate logică care este accesată exclusiv de unul dintre procesele curente.

O astfel de secvență de instrucțiuni este o secțiune critică logică. Deoarece procesul care a intrat în secțiunea sa critică exclude de la execuție orice alt proces curent, rezultă că are loc o excludere reciprocă (mutuală) a proceselor între ele.

Dacă execuția secțiunii critice a unui proces impune existența unei zone de memorie pe care acel proces să o monopolizeze pe durata execuției secțiunii sale critice, atunci acea zonă de memorie devine o secțiune critică fizică sau resursă critică. De cele mai multe ori, o astfel de secțiune critică fizică este o zonă comună de date partajate de mai multe procese.

Problema excluderii mutuale este legată de proprietatea secțiunii critice de a fi atomică, indivizibilă la nivelul de bază al execuției unui proces. Această atomicitate se poate atinge fie valorificând facilitățile primitivelor limbajului de programare gazdă, fie prin mecanisme controlate direct de programator (din această categorie fac parte algoritmii specifici pentru rezolvarea anumitor probleme concurente).

Problema excluderii mutuale apare deoarece în fiecare moment cel mult un proces poate să se afle în secțiunea lui critică. Potrivit lui *Andrews* [5], excluderea mutuală este o primă formă de sincronizare, ca o alternativă pentru sincronizarea pe condiție.

Problema excluderii mutuale este una centrală în contextul programării concurente. Rezolvarea problemei excluderii mutuale depinde de îndeplinirea a trei cerințe fundamentale și anume [6]:

- excluderea reciprocă propriu-zisă - la fiecare moment de timp cel mult un proces se află în secțiunea sa critică;
- competiția constructivă, neantagonistă - dacă nici un proces nu este în secțiunea critică și dacă există procese care doresc să intre în secțiunile lor critice atunci unul dintre acestea va intra efectiv;
- conexiunea liberă între procese - dacă un proces "întârzie" în secțiunea sa necritică atunci această situație nu trebuie să împiedice alt proces să intre în secțiunea sa critică (dacă dorește acest lucru).

Îndeplinirea celei de-a doua condiții asigură faptul că procesele nu se împiedică unul pe altul să intre în secțiunea critică și nici nu se invită la nesfârșit unul pe altul să intre în secțiunile critice.

Cea de-a treia condiție asigură faptul că, dacă "întârzierea" este definitivă, aceasta nu duce la blocarea întregului sistem (cu alte cuvinte, blocarea locală nu implică blocarea globală a sistemului).

Excluderea reciprocă (mutuală) fiind un concept central, esențial al concurenței, în paragrafele următoare vom reveni asupra rezolvării acestei probleme la diferite niveluri de abstractizare și folosind diferite primitive. Principalele abordări sunt:

- folosirea suportului *hardware* al sistemului gazdă: rezolvarea excluderii mutuale se face cu ajutorul *TAS*-urilor (*Test And Set*); aceste primitive testează și setează variabile *booleene* (o astfel de variabilă este asociată unei resurse critice); un *TAS* nu este o operație atomică;

- folosind arbitrajul memoriei comune: soluția vizează secțiunile critice și nu resursele critice (cum s-a lucrat la nivelul *TAS*-urilor); la acest nivel aplicațiile folosesc semafoare;
- folosind suportul *software*: rezolvarea excluderii mutuale se bazează pe facilitățile oferite de limbajele de programare pentru comunicarea și sincronizarea proceselor executate nesecvențial.

Interblocarea

Se spune că un grup de procese concurente sunt ***interblocate*** dacă fiecare deține resurse comune care trebuie să fie alocate de alte procese pentru a-și continua execuția [2].

Interblocarea poate apărea ori de câte ori mai multe procese partajează resurse într-o manieră nesupervizată. De aici rezultă că interblocarea poate să apară atât în sisteme care operează în multiprogramare, cât și în sistemele multicalculator și multiprocesor.

Când un proces își alocă o resursă se spune că blochează acea resursă. Operațiile **LOCK** și **UNLOCK** corespund operatorilor *P* și *V* definiți de *Dijkstra* pentru semafoarele binare (vezi episodul următor) [7].

Exemple

Într-un sistem multiprocesor poate apărea următoarea situație de interblocare în care sunt implicate două procese, *Proces* 1 și *Proces* 2 și două resurse, *A* și *B*:

Proces 1	*Proces* 2
...	...
`LOCK(A)`	`LOCK(B)`
...	...
`LOCK(B)`	`LOCK(A)`
...	...

Atâta timp cât fiecare proces are acces exclusiv la o resursă cerută (solicitată) de celălalt proces, nici unul nu-și poate continua activitatea.

Într-un sistem multicalculator poate să apară un caz particular de interblocare și anume cea de tip *buffer deadlock*. Considerăm un sistem multicalculator în care procesele comunică sincron. Aceasta înseamnă că atunci când un proces *E* trimite un mesaj către un alt proces *D*, expeditorul *E* se blochează în așteptarea confirmării de la destinatarul *D*. În această situație, la expediere, mesajul este depus într-un *buffer* de mesaje ale destinatarului.

Presupunem că mai multe procese au trimis mesaje către *D*, acesta nu le-a preluat la timp și *buffer*-ul este plin. Cât timp *D* nu citește mesaje (deci nu-și descarcă *buffer*-ul), nici un mesaj trimis către *D* nu va mai putea fi recepționat și toate procesele expeditor sunt blocate în așteptarea confirmării.

Fie *I* un proces care încearcă să trimită mesaje către *D* cât timp *buffer*-ul este plin. Dacă *D* așteaptă să citească ***cu prioritate*** mesajul de la *I*, atunci procesul *D* se va bloca în așteptarea mesajului de la *I*. La rândul lui, procesul *I* este deja blocat în așteptarea confirmării de la *D*. Rezultă că cele două procese sunt interblocate datorită umplerii *buffer*-ului de recepționare a mesajelor.

În general, interblocarea poate apărea dacă:
- sistemul trebuie să asigure excluderea mutuală a proceselor la accesul pe resurse comune;
- sistemul nu respectă condiția de *preemption* (previziune, prioritate): un proces nu eliberează resursele pe care le deține dacă nu le-a utilizat pe toate;
- sistemul trebuie să respecte condiția de *resource waiting*: fiecare proces deține resursele cât timp așteaptă ca alte procese să le elibereze pe ale lor;
- există un lanț de procese în așteptare: fiecare proces așteaptă să se elibereze resursele pe care le deține un alt proces din lanț.

Indiferent de tipul de sistem nesecvențial, acesta trebuie să fie prevăzut cu mecanisme specifice pentru detectarea interblocării. Cu alte cuvinte, interblocarea este o stare a sistemului care trebuie evitată.

Rezolvarea problemei interblocării poate fi privită în trei moduri:
- prima abordare este să se detecteze apariția interblocării și să se încerce refacerea contextului anterior interblocării;
- a doua abordare este să se evite interblocarea prin utilizarea ***în avans*** a informațiilor referitoare la cererile de resurse, astfel încât să se realizeze un control al alocării resurselor care să ***preîntâmpine*** apariția interblocării;
- a treia abordare este să se prevină interblocarea prin a interzice apariția ultimelor trei situații prezentate anterior.

Bibliografie

1. Bacon J., *Concurrent Systems - operating systems, database and distributed systems: an integrated approach*, Addison-Wesley, 1998
2. Quinn M.J., *Parallel computing - theory and practice*, McGraw-Hill, 1994
3. Braunl T., *Parallel programming - an introduction*, Prentice-Hall, 1993
4. Ionescu F., *Principiile calculului paralel*, Editura Tehnică București, 1999
5. Andrews G.R., *Concurrent Programming - Principles and Practice*, Redwood City CA: Benjamin/Cummings, 1991
6. Georgescu H., *Programare concurentă - teorie și aplicații*, Editura Tehnică București, 1996
7. Dijkstra E.W,. *The structure of THE operating system*, Comm. ACM 11(5), 1968

D-na Anca Vasilescu este asistent universitar la Universitatea Transilvania din Brașov și poate fi contactată prin e-mail la adresa **vasilex@info.unitbv.ro**. *D-na Oana Georgescu este asistent universitar la Universitatea Sextil Pușcariu din Brașov și poate fi contactată prin e-mail la adresa* **o.georgescu@info.unitbv.ro**.

Tehnici de CRIPTARE

Claudiu Soroiu

În acest număr continuăm prezentarea tehnicilor de criptare cu descrierea metodelor S-DES și DES standard, tehnicile de criptare cu cea mai largă utilizare.

Simple Data Encryption Standard

Tehnica de criptare *DES* (*Data Encryption Standard*) a apărut în anii '70 în *Statele Unite ale Americii*. Standardul de criptare a fost realizat de *Biroul Național de Standardizare* (*National Bureau of Standards*) cu ajutorul *Agenției Naționale de Securitate* (*National Security Agency*). Această tehnică de criptare este una dintre cele mai utilizate. Scopul acestei tehnici este acela de a asigura o metodă standard pentru protejarea datelor comerciale neclasificate. *IBM* a creat în anul 1976 o primă implementare a algoritmului sub numele de *Lucifer*.

Acest algoritm utilizează reprezentarea binară a codurilor numerice ale caracterlor folosite pentru reprezentarea datelor care urmează a fi criptate.

Algoritmul de bază *DES* folosește blocuri de 64 de biți de date și aplică doar două operații asupra intrării: deplasare și substituție de biți. Cheia de criptare controlează procesul de codificare a datelor. Prin aplicarea celor două operații în mod repetat și neliniar, se obține un rezultat care, practic, nu poate fi decriptat decât dacă se cunoaște cheia de criptare.

Asupra fiecărui bloc de date, algoritmul se aplică de până la 16 ori (*DES* standard aplică algoritmul de criptare de 16 ori).

Datorită complexității ridicate a algoritmului *DES* standard, în continuare vă prezentăm varianta simplificată a algoritmului *DES*, *Simple-DES*, care folosește blocuri de 8 biți de date.

Algoritmul de criptare

Algoritmul *DES simplificat* poate fi exprimat ca o compunere de cinci funcții:

- o permutare inițială *IP*, care permută biții blocului de intrare;
- o funcție *f*, care depinde de o subcheie k_1 a unei chei de criptare *K*; funcția este foarte complexă și acționează asupra jumătății stângi a blocului de date rezultat la pasul anterior;
- o permutare *SW*, care interschimbă cele două jumătăți ale blocului de date rezultat la pasul anterior. Dacă blocul de date este $d = d_1d_2d_3d_4d_5d_6d_7d_8$, atunci acesta devine $d_5d_6d_7d_8d_1d_2d_3d_4$, unde d_i reprezintă al *i*-lea bit din blocul *d*;
- aceeași funcție *f*, care folosește o altă subcheie (k_2) a cheii de criptare *K*;
- inversa permutării *IP*, IP^{-1}.

Formula matematică ce stă la baza algoritmului de criptare este:

$$Y = E_k(X) = IP^{-1}(f_{k_2}(SW(f_{k_1}(IP(X))))),$$

unde E_K este funcția de criptare care folosește cheia de criptare *K* și se aplică asupra blocului de intrare *X*, rezultând blocul criptat *Y*; f_{k_1} și f_{k_2} reprezintă funcția *f* aplicată asupra unui bloc de date, folosind subcheile k_1, respectiv k_2.

În cazul acestui algoritm fiecare bloc de intrare are 8 biți și cheia de criptare, care este partajată între emițător și receptor, are lungimea de 10 biți. Cheia de criptare este folosită pentru a crea două subchei având fiecare 8 biți.

La început se construiesc cele două subchei de criptare k_1 și k_2. Algoritmul este prezentat în cele ce urmează.

Considerăm cheia de criptare $b = b_1b_2b_3b_4b_5b_6b_7b_8b_9b_{10}$.

Asupra cheii de criptare *b*, se aplică o permutare și se obține un bloc *c* care are 10 biți.

$$c = c_1c_2c_3c_4c_5c_6c_7c_8c_9c_{10} = b_3b_5b_2b_7b_4b_{10}b_1b_9b_8b_6.$$

Cele două jumătăți ale blocului *c* sunt deplasate circular la stânga cu 1 bit și componentele rezultate sunt folosite pentru a construi un bloc *d* care are, de asemenea, 10 biți:

$$c_1c_2c_3c_4c_5 \rightarrow c_2c_3c_4c_5c_1$$
$$c_6c_7c_8c_9c_{10} \rightarrow c_7c_8c_9c_{10}c_6$$
$$c_2c_3c_4c_5c_1c_7c_8c_9c_{10}c_6 = d$$

La pasul următor, ultimii 8 biți ai blocului *d* sunt permutați și se obține prima subcheie, k_1.

$$d_3d_4d_5d_6d_7d_8d_9d_{10} \rightarrow d_6d_3d_7d_4d_8d_5d_{10}d_9 = k_1$$

În continuare, cele două jumătăți ale blocului *d* sunt deplasate circular la stânga cu două poziții, și rezultatul este folosit pentru a obține un bloc de date *e*, care are 10 biți.

$$d_1d_2d_3d_4d_5 \rightarrow d_3d_4d_5d_1d_2$$
$$d_6d_7d_8d_9d_{10} \rightarrow d_8d_9d_{10}d_6d_7$$
$$d_3d_4d_5d_1d_2d_8d_9d_{10}d_6d_7 = e$$

Ultimii 8 biți ai blocului e sunt permutați folosind aceeași permutare (cea utilizată și pentru ultimii 8 biți ai blocului d), iar rezulatul obținut reprezintă cea de-a doua subcheie k_2.

$$e_3e_4e_5e_6e_7e_8e_9e_{10} \rightarrow e_6e_3e_7e_4e_8e_5e_{10}e_9 = k_2$$

Pentru a construi funcția de criptare f, considerăm că blocul de intrare de 8 biți, folosit de aceasta, este format din două părți (stângă - L și dreaptă - R) de câte patru biți fiecare. Cu această condiție, funcția f este:

$$f_k(L, R) = (L \oplus F(R, k), R),$$

unde $\oplus$ reprezintă operatorul binar *SAU-exclusiv*, și $L \oplus F(R, k)$ împreună cu R formează blocul de ieșire de 8 biți. k reprezintă o subcheie de criptare de 8 biți. Funcția F primește ca parametri un bloc de date de 4 biți și o subcheie k, și furnizează la ieșire un bloc de 4 biți.

Se observă că funcția f modifică numai partea stângă a blocului de intrare, urmând ca după aplicarea funcției *SW* să se modifice și cealaltă jumătate a blocului.

În continuare vom descrie funcția F.

Fie $R = n_1n_2n_3n_4$. Asupra lui R se aplică două permutări și rezultatele celor două permutări formează un bloc de 8 biți, această operație purtând numele de *expandare*.

$$n_1n_2n_3n_4 \rightarrow n_4n_1n_2n_3$$
$$n_1n_2n_3n_4 \rightarrow n_2n_3n_4n_1$$
$$n_4n_1n_2n_3,\ n_2n_3n_4n_1 \rightarrow n_4n_1n_2n_3n_2n_3n_4n_1$$

Blocul de 8 biți se va scrie astfel:

$$n_4 \mid n_1\ \ n_2 \mid n_3$$
$$n_2 \mid n_3\ \ n_4 \mid n_1$$

La pasul următor se aplică, asupra acestui bloc, cheia de criptare k astfel:

$$n_4 \oplus k_1 \mid n_1 \oplus k_2\ \ n_2 \oplus k_3 \mid n_3 \oplus k_4 \rightarrow p_{00} \mid p_{01}\ \ p_{02} \mid p_{03}$$
$$n_2 \oplus k_5 \mid n_3 \oplus k_6\ \ n_4 \oplus k_7 \mid n_1 \oplus k_8 \rightarrow p_{10} \mid p_{11}\ \ p_{12} \mid p_{13}$$

În continuare se definesc două matrice S_1 și S_2 numite și ***S-boxes*** (*substitution boxes* - matrice de substituție), astfel:

S_0 =

	0	1	2	3
0	1	0	3	2
1	3	2	1	0
2	0	2	1	3
3	3	1	0	2

S_1 =

	0	1	2	3
0	0	1	2	3
1	2	0	1	3
2	3	0	1	2
3	2	1	0	3

Cele două matrice de substituție se folosesc astfel:

- biții p_{00} și p_{03} formează un număr cuprins între 0 și 3 care reprezintă numărul unei linii l_0 din matricea S_0;
- biții p_{10} și p_{13} formează un număr cuprins între 0 și 3 care reprezintă numărul unei linii l_1 din matricea S_1;
- biții p_{01} și p_{02} formează un număr cuprins între 0 și 3 care reprezintă numărul unei coloane c_0 din matricea S_0;
- biții p_{11} și p_{12} formează un număr cuprins între 0 și 3 care reprezintă numărul unei coloane c_1 din matricea S_1;
- numărul de pe poziția l_0, c_0 din matricea S_0 se scrie în baza 2 și se obțin 2 biți m_1 și m_2;
- numărul de pe poziția l_1, c_1 din matricea S_1 se scrie în baza 2 și se obțin 2 biți m_3 și m_4;
- în final se obține blocul de 4 biți $m_1m_2m_3m_4$ căruia i se aplică o permutare și devine $m_2m_4m_3m_1$.

Algoritmul de decriptare este dat de inversa funcției E_k.

Datorită faptului că algoritmul *S-DES* folosește chei de criptare care au lungimea de 10 biți, există doar 1024 posibile chei de criptare. Așadar, un text criptat folosind această metodă poate fi decriptat, în eventualitatea în care nu se cunosc cheile de criptare, folosind un atac cu text necriptat și fiecare cheie de criptare posibilă sau dicționare de chei sau de parole.

Mai mult de 20 de ani s-a crezut că algoritmii de criptare *DES standard* și alte variante ale acestuia sunt atât de siguri încât mesajele criptate cu acesta nu pot fi decriptate decât dacă se cunosc cheile de criptare.

Algoritmul de criptare DES complet

Algoritmul *DES* standard folosește ca intrare blocuri de 64 de biți și chei de criptare de 56 de biți (ceea ce înseamnă că există 2^{56} = 72.057.594.037.927.936 de chei posibile distincte). La baza acestui algoritm se află următoarea formulă matematică:

$$Y = IP^{-1} \circ f_{k_{16}} \circ SW \circ f_{k_{15}} \circ SW \circ ... \circ SW \circ f_{k_1} \circ IP(X)$$

O cheie de criptare K este folosită pentru a crea 16 subchei k_1, k_2, ..., k_{16} având câte 48 de biți.

Funcția F, care intră în componența funcției f_k, are ca parametri un bloc de 32 de biți și o subcheie de criptare. După cum se observă, unui bloc de date, care a trecut de permutarea inițială, îi este aplicată funcția f_k de 16 ori.

Pașii care trebuie aplicați de 16 ori consecutiv, pentru a cripta un bloc de 64 de biți, sunt următorii:

- unui bloc din mesajul inițial îi este aplicată permutarea inițială și apoi este împărțit în două jumătăți L_0 și R_0;
- se generează cele 16 subchei din cheia de criptare K, prin aplicarea unei permutări inițiale urmată de 16 deplasări circulare la stânga cu 1 bit ale blocului rezultat, fiecare deplasare fiind urmată de o permutare a ultimilor 48 de biți și rezultând, la fiecare pas i, o subcheie de criptare k_i;
- algoritmul pentru funcția de criptare f_k este următorul:
  - la fel ca în cazul algoritmului *S-DES*, la fiecare pas i avem:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, k_i)$$

  - funcția F are doi parametri: blocul R_{i-1} care are 32 de biți și subcheia k_i. Cei 32 de biți sunt expandați la 48 după care sunt permutați și se aplică operatorul *SAU-exclusiv* între aceștia și subcheia k_i. Rezultatul este împărțit în 6 blocuri de câte 8 biți, fiecare fiind prelucrat folosindu-se 8 *S-matrice* distincte. Pentru cele 8 *S-matrice* intrarea are 6 biți, iar la ieșire sunt furnizați 4. Fiecare *S-matrice* are 4 linii și 16 coloane și fiecare linie conține o permutare distinctă a șirului format din numerele de la 0 la 15. Primul și ultimul bit din fiecare grup de 6 biți sunt folosiți pentru a

identifica numărul liniei din *S-matricea* corespunzătoare, iar ceilalți 4 sunt folosiți pentru numărul coloanei. Toate grupurile de 4 biți obținute din *S-matrice* în același mod ca și în cazul metodei *S-DES* formează un bloc de 32 de biți, ai cărui biți sunt în final permutați și reprezintă rezultatul funcției *F*;

- ♦ între rezultatul funcției *F* și jumătatea stângă L_{i-1} se aplică operatorul *SAU-exclusiv*;

- cu ajutorul funcției *SW*, cele două jumătăți ale blocului rezultat se interschimbă:

$$(L_i, R_i) \rightarrow (R_i, L_i)$$

- la sfârșitul celor 16 iterații, cele două jumătăți rezultate formează un număr asupra căruia se aplică inversa permutării inițiale, obținându-se blocul criptat.

Detalii

Algoritmul de criptare *DES* este un caz particular al clasei de tehnici de criptare numite criptări *Feistel*, care folosesc mai multe blocuri de date. Aceste tehnici folosesc alternativ mai multe funcții de *difuzie* și *confuzie*.

Funcțiile de difuzie au ca scop construirea de relații statistice între mesajul inițial și cel criptat, cât de complex este posibil cu scopul de a proteja mesajul împotriva atacurilor statistice care au ca scop construirea cheii de criptare. În procesul de criptare folosind blocuri de date, acest lucru se realizează prin permutarea elementelor blocului înainte de aplicarea unei funcții.

Funcțiile de confuzie au ca scop construirea de relații statistice între mesajul criptat și cheia de criptare, care să fie cât de complexe posibil. Aceasta se realizează prin utilizarea unui algoritm complex de substituție.

Un algoritm *Feistel* folosește blocuri de intrare având lungimea un număr par care este împărțit în două jumătăți L_i și R_i, și o subcheie k_1, derivată dintr-o cheie de criptare *K*. Cele două jumătăți sunt procesate de *n* ori, după care sunt recombinate pentru a forma blocul de ieșire (criptat). Fiecare operație de procesare are aceeași structură.

Un bloc *L* este substituit cu rezultatul unei expresii de forma $L \oplus F(R, k)$. După substituție, cele două jumătăți se interschimbă.

Algoritmii de criptare *Feistel* depind de:

- *mărimea blocului de intrare*: cu cât blocul de intrare este mai mare, cu atât securitatea datelor crește, dar crește și timpul alocat de calculator pentru procesare; de obicei se folosesc blocuri de 64 de biți;
- *mărimea cheii de criptare*: cu cât cheia de criptare este mai mare, cu atât securitatea datelor crește, dar crește și timpul necesar pentru prelucrare; de obicei se folosesc blocuri de 64 de biți;
- *numărul de iterații*: de obicei se folosesc 16 iterații în procesul de criptare; acest număr reprezintă un compromis rezonabil între securitate și viteză de procesare;
- *generarea subcheilor*: cu cât complexitatea subcheilor generate este mai mare, cu atât crește securitatea datelor;
- *funcția F*: această funcție este foarte importantă, deoarece, în general, aceasta nu este o funcție liniară și cu cât este mai complexă, cu atât securitatea datelor este mai mare.

După cum se observă, creșterea complexității oricăreia dintre cele 5 componente enumerate mai sus duce la creșterea timpului de procesare necesar pentru un calculator.

O facilitate interesantă și foarte utilă a algoritmilor de criptare *Feistel* este aceea că algoritmul de decriptare este același cu cel de criptare, singura diferență fiind aceea că subcheile obținute din cheia de criptare sunt folosite în ordine inversă.

Pentru algoritmul *DES* se folosesc diferite modele pentru a asigura diferite tipuri de securitate. Cel mai simplu model poartă numele de *ECB* (*Electronic CookBook*). Acest model criptează fiecare bloc de 64 de biți de date independent, folosind aceeași cheie. În consecință, aceleași două blocuri de date identice sunt criptate la fel, folosind acest model. Modelul *ECB* este folosit, de obicei, pentru a cripta un singur mesaj scurt care poate conține, de exemplu, o cheie de criptare.

Cel de-al doilea model este *CBC* (*Cipher Block Chaining*). Fie P_i și C_i ale *i*-lea blocuri de date necriptate și criptate și E_k funcția de criptare cu cheia *k*, atunci:

$$C_1 = E_k(P_1 \oplus IV),\ C_n = E_k(C_{n-1} \oplus P_n),\ n \geq 2,$$

unde *IV* este un vector de inițializare care trebuie protejat în același mod în care este protejată și cheia de criptare. Cu ajutorul acestui model, două blocuri de date de intrare identice sunt criptate diferit. Acest model este folosit pentru a cripta mesaje lungi de date.

Dacă notăm cu D_k funcția de decriptare care folosește cheia *k*, atunci formula de decriptare a unui mesaj este:

$$C_{n-1} \oplus D_k(C_n) = C_{n-1} \oplus D_k(E_k(C_{n-1} \oplus P_n)) = C_{n-1} \oplus C_{n-1} \oplus P_n = P_n.$$

Cu toate că *DES* este un algoritm care criptează blocuri de date, acesta poate fi trasformat într-un algoritm care criptează fluxuri de date, folosind *CFM* (*Cipher Feedback Mode*).

Să presupunem că dorim să transmitem un flux de caractere criptat care constă în *j* biți pe caracter (de obicei se folosesc 8 biți).

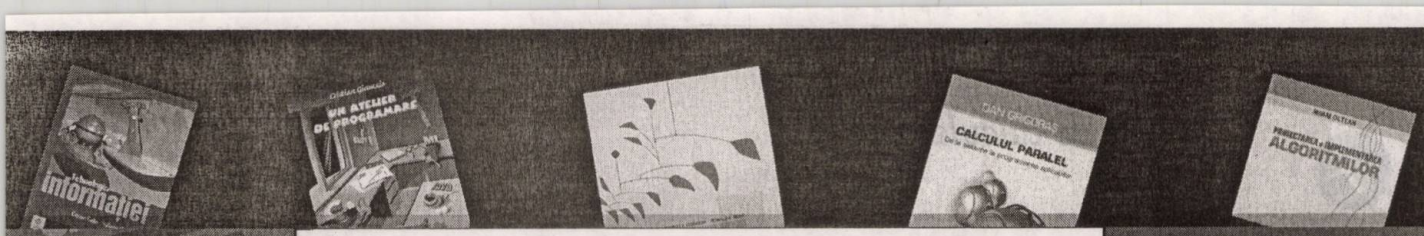
Dacă avem un vector *IV* de inițializare cu lungimea de 64 de biți, $S_j(B)$ sunt cei mai din stânga *j* biți ai blocului *B* și P_i, C_i au semnificația de mai sus, atunci fiecare caracter criptat este obținut astfel:

$$C_i = P_i \oplus S_j(E_k(B_i)),\ B_i = (B_{i-1}[(j+1):64], C_{i-1}),\ B_1 = IV,$$

unde $(B[(j+1):64], C)$ reprezintă un bloc de 64 de biți format prin deplasarea circulară la stânga cu *j* biți a blocului *B*, după care ultimii *j* biți din blocul B sunt înlocuiți cu blocul *C* (numit *registru de deplasare cu j biți*).

Operația de decriptare a unui astfel de mesaj este similară procesului de criptare, singura deosebire fiind aceea că se inversează rolurile lui C_i și P_i, deci P_i se transformă în C_i și invers.

Claudiu Soroiu este redactor al GInfo. Poate fi contactat prin e-mail la adresa csoroiu@yahoo.com.



COMPUTER LIBRIS AGORA

Cartea este o culegere de studii de caz, construite pe temelia problemelor concursurilor Southeastern European Regional Programming Contest din perioada 1995-1999, concursuri organizate sub egida Association for Computing Machinery.
Cartea se adresează atât începătorilor, cât și celor cu experiență în domeniul programării, în încercarea de a contura prin exemple imaginea unei lumi diverse și pasionante, dar pretențioase, aceea a înzestrării mașinii de calcul cu capacitatea de a rezolva corect și eficient.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

Rețele ETHERNET

Mihai Scorțaru

Fără îndoială, cel mai răspândit tip de rețele locale folosit în prezent este Ethernet. În cadrul acestui episod vom prezenta câteva detalii referitoare la rețelele locale construite pe baza acestui standard.

Medii de transmisie

În anii '70 trei companii producătoare de echipamente *hardware* au format consorțiul ***DIX*** pentru a dezvolta o rețea de tip *LAN*. Denumirea consorțiului provenea de la numele celor trei companii: *Digital Equipment*, *Intel* și *Xerox*.

Primele rezultate semnificative au apărut la începutul anilor '80 când a fost redactată o specificație pentru rețele locale ***Ethernet v1.0***. După câțiva ani a apărut o nouă specificație, numită ***Ethernet v2.0***, care stă la baza definirii conceptelor referitoare la rețelele locale.

Pe baza acestei specificații a fost redactat standardul *IEEE* 802.3, care a fost adoptat ulterior și de către *ISO* prin elaborarea standardului *ISO* 8802.3.

Acest standard diferă puțin de specificația originală, mai ales prin câteva aspecte de natură logică (la nivelul *legăturii de date*) sau electrică (la nivelul *fizic*).

La nivelul fizic, standardul indică o topologie de tip magistrală care trebuie realizată folosind cablu coaxial și funcționează la o viteză de 10 *MBps*.

La nivelul *MAC*, este implementată o metodă de acces la mediu, denumită ***CSMA/CD*** (*Carrier Sense Multiple Access / Collision Detection*).

Metoda CSMA/CD

După cum arată și denumirea, această metodă de acces operează în trei faze:

- ***sesizarea purtătoarei*** (*carrier sense*) - fiecare stație care dorește să inițieze o transmisie de date trebuie mai întâi să "asculte" magistrala pentru a verifica dacă este ocupată; acest principiu este denumit ***"ascultă înainte de a vorbi"*** (*listen before talking*);
- ***accesul multiplu*** (*multiple access*) - fiecare stație care a detectat că magistrala este liberă poate să inițializeze o transmisie; dacă două sau mai multe stații încep să transmită simultan atunci, datorită faptului că timpul de propagare a semnalului nu este nul, va apărea o ***coliziune***; aceasta va afecta toate semnalele de date și transmisia va fi compromisă;
- ***detectarea coliziunii*** (*collision detection*) - fiecare stație care transmite date poate detecta coliziunile deoarece stațiile ascultă linia în timpul transmisiei; principiul este denumit ***"ascultă în timp ce vorbești"*** (*listen while talking*).

Prima stație care detectează o conexiune își va suspenda emisia și va trimite un semnal special care anunță existența unei ***interferențe*** (*jamming*). Acest semnal va avea o frecvență specială și o durată echivalentă unei transmisii de 32 de biți. Toate stațiile vor detecta acest semnal și își vor suspenda activitatea.

Pentru a micșora șansele de apariție a unei noi coliziuni când stațiile vor începe din nou să transmită, timpul care va trece până la reluarea transmisiilor va fi diferit pentru fiecare stație. Intervalele de timp sunt stabilite pe baza unui ***algoritm de revenire*** (*back-off algorithm*).

Metoda de acces *CSMA/CD* are o eficiență teoretică de 30%. Practic, s-a observat că ea funcționează optim în cazul în care factorul de încărcare a rețelei este de 30% (viteza este de 3 *Mbps*) și există vârfuri de funcționalitate optimă până la viteze de 6 *Mbps*.

Un parametru foarte important pentru funcționarea rețelei este ***întârzierea de propagare a coliziunii*** (*round trip collision delay*). Pentru ca algoritmul să funcționeze corect trebuie ca stațiile să nu își termine transmisiile înainte de a putea detecta coliziunea. Așadar, stațiile trebuie să detecteze în timp util coliziunile, indiferent de lungimea cadrului care este transmis și de distanța maximă dintre două stații. ***Cazul critic*** este dat de cea mai defavorabilă situație care poate apărea:

- cele două stații implicate în transmisiei (emițătorul și receptorul) se află la cele două extremități ale rețelei;
- emițătorul inițializează transmiterea unui cadru către receptor;
- când cadrul se află foarte aproape de receptor, acesta inițiază, la rândul său, o transmisie;
- coliziunea are loc în imediata vecinătate a receptorului;
- semnalul de coliziune se propagă până la emițător;
- emițătorul detectează coliziunea și oprește transmisia.

Întârzierea de propagare a coliziunii (determinată prin însumarea timpului necesar ca semnalul de date să parcurgă traseul de la emițător la receptor și a timpului necesar ca semnalul de coliziune să parcurgă traseul de la receptor la emițător) a fost standardizată la 49,9 μs.

Un pachet care are lungimea minimă de 512 bytes care este transmis cu o viteză de 10 MHz necesită un timp de 51,2 μs, așadar este îndeplinită condiția ca timpul de transmitere a cadrului să fie mai mare decât întârzierea de propagare a coliziunii.

Este evident că distanța maximă dintre două stații nu poate fi mai mare decât o anumită valoare D_{max}. Această valoare este calculată astfel: se consideră că semnalul de date se propagă între emițător și receptor în 22,8 μs; semnalul de coliziune are alți parametri, iar timpul său de propagare până la emițător se consideră a fi de 27,1 μs (prin însumarea celor două valori se obține valoarea de 49,9 μs care, așa cum am afirmat anterior, este standardizată pentru întârzierea de propagare a coliziunii); distanța maximă este dată de produsul dintre viteza semnalului (se alege o valoare acceptabilă de $0{,}66 \cdot c \approx 200.000$ km/s) și timpul de propagare a semnalului într-un sens al rețelei; se obține valoarea $D_{max} \approx 4{,}5$ km.

Operațiile care trebuie să poată fi efectuate dacă se implementează această metodă de acces la mediu sunt:

- *transmisia de pachete* (*cadre*) - subnivelul *MAC* acceptă un pachet de la nivelul superior și formează un flux de biți care este trimis către mediul fizic de comunicație;
- *recepția de pachete* (*cadre*) - subnivelul *MAC* recepționează un flux de biți de nivelul fizic și formează un pachet pe care îl trimite nivelului superior; în cazul apariției unor erori de adresare pachetul este pierdut;
- *așteptarea pentru inițializarea transmisiei* - transmisia nu poate începe înaintea eliberării canalului;
- *generarea și verificarea sumei de control* - pentru cadrele primite trebuie verificat câmpul care conține suma de control, iar pentru cadrele trimise trebuie generat un astfel de câmp;
- *spațierea temporală* (*inter-frame spacing*) - pentru recunoașterea sfârșitului de cadru și separarea sa de cadrul următor trebuie să existe un timp minim de așteptare înainte să înceapă transmiterea următorului cadru;
- *planificarea retransmisiei* - timpul de așteptare după detectarea coliziunilor trebuie stabilit pe baza algoritmului de regresie;
- *emiterea semnalului de detectare a coliziunii* - în cazul în care se detectează o coliziune trebuie întreruptă transmisia, după care este emis semnalul de *jamming*;
- *verificarea lungimii cadrelor recepționate* - trebuie verificat faptul că un cadru primit are o lungime mai mare sau egală cu cea minimă (64 bytes) și mai mică sau egală cu cea maximă;
- *generarea și eliminarea preambulului* - în momentul emiterii unui cadru trebuie generat un câmp preambul cu rol în sincronizare; în momentul în care se primește un cadru, câmpul preambul al acestuia trebuie eliminat.

Standardul Ethernet v2.0

Acesta este un standard "de facto" folosit fără a fi redactat de nici o organizație de standardizare. Există unele diferențe față de standardele din familia 802.x; standardul *Ethernet* acoperă nivelurile fizic și legătură de date ale standardului *OSI*.

Nivelul fizic

Rețelele *Ethernet* folosesc codificarea *Manchester* și operează la viteze de 10 *Mbps* și frecvență de minimum 20 MHz.

Potrivit standardului, distanța maximă dintre două stații este de 2,8 km, numărul maxim de stații este de 1028, mediul de transmisie este cablul coaxial gros de tip RG-213 cu impedanța de 50Ω, iar topologia rețelei este de tip magistrală.

Stațiile conectate la anumite distanțe una față de cealaltă folosind un cablu de lungime limitată formează un ***segment***. Pentru a extinde lungimea rețelei și numărul de stații segmentele pot fi interconectate prin intermediul unor ***repetoare active***. Acestea operează la nivel fizic și au rolul de a regenera semnalele de date sau de coliziune. ***Lungimea*** rețelei este dată de suma tuturor segmentelor de cablu.

Diametrul cablului folosit pentru cablare trebuie să fie de 0,4 inci (aproximativ 1 cm), viteza de propagare a semnalului trebuie să fie de minimum $0{,}77 \cdot c$, iar atenuarea maximă a unui segment trebuie să fie de cel mult 8,5 dB.

Un dezavantaj demn de luat în seamă este rigiditatea cablurilor. Acestea nu pot fi îndoite pentru a ajunge în apropierea stațiilor de lucru; de obicei este folosită cablarea pe perete. Inițial, cablul *Ethernet* era galben și era marcat la intervale de 2,5 m pentru a indica locul în care ar trebui să se amplaseze interfețele pentru conectare.

La capetele fiecărui segment trebuie atașați terminatori cu impedanța de 50Ω pentru a elimina interferențele din cabluri; unul dintre cei doi terminatori trebuie legat la pământ.

Nivelul legăturii de date

Un cadru *Ethernet* conține 1536 de bytes, grupați în șapte câmpuri.

Primul câmp al pachetului este ***preambulul***, are o lungime de șapte bytes și este folosit, în special, pentru sincronizarea ceasului receptorului cu ceasul emițătorului.

Al doilea câmp este ***SFD*** (*Start Frame Delimiter*) și are lungimea de un byte. El conține biți ***non-date*** care au o codificare *Manchester* diferită de biții care reprezintă valorile 0 și 1, neexistând tranziții la mijlocul perioadei de bit. Notațiile consacrate pentru astfel de biți sunt *J* și *K*.

Următoarele două câmpuri sunt *SA* (*Source Address*) și *DA* (*Destination Address*) și reprezintă adresele *MAC* ale emițătorului, respectiv receptorului, având fiecare o lungime de șase bytes.

Câmpul următor este *Tip*, are o lungime de doi bytes și conține un cod al protocolului de nivel superior care este destinatarul sau generatorul unității de date conținute.

Câmpul *Info* conține unitatea de date propriu-zisă și are lungimea de 1454 de bytes.

Ultimul câmp este *FCS* (*Frame Control Sequence*), are o lungime de 32 de bytes și reprezintă suma de control calculată pentru câmpurile anterioare.

Se observă că nu există un câmp de sfârșit de cadru, rolul său fiind preluat de intervalul dintre cadre. Potrivit standardului, acest interval este de 9,6 μs.

Componentele rețelelor Ethernet

Există trei elemente principale care sunt folosite pentru crearea rețelelor de tip *Ethernet*:

- interfața *Ethernet - controller*-ul de rețea;
- *transceiver*-ul - elementul de atașare la mediu;
- cablul pentru *transceiver*.

Interfața este modulul care asigură conectarea stației de lucru la *transceiver*. Astfel se realizează legătura între magistrala internă a calculatorului și elementul de atașare la mediu. Principalele funcții ale acestei componente sunt:

- asigurarea interfeței dintre magistralele interne și *transceiver*;
- asamblarea și dezasamblarea cadrelor;
- controlul adresării;
- controlul erorilor;
- încapsularea datelor în formatul cadrelor *Ethernet*;
- decapsularea datelor din formatul cadrelor *Ethernet*;
- gestionarea legăturii (implementarea algoritmului de acces la mediu);
- transformarea datelor codificate prin metoda *Manchester* în date codificate prin metoda *Non-Return to Zero*;
- transformarea datelor codificate prin metoda *Non-Return to Zero* în date codificate prin metoda *Manchester*.

Transceiver-ul este elementul care permite atașarea la mediu, realizând legătura dintre acesta și interfață. Aceasta din urmă este conectată la transceiver prin intermediul unui așa-numit ***cablu transceiver***.

Principalele funcții ale acestei componente sunt:

- transmiterea către mediu a datelor primite de la interfață; această transmitere se realizează prin intermediul unor circuite de formare a semnalelor electrice;
- recepționarea datelor care sosesc din mediu;
- recepționarea semnalelor de coliziune din mediu; în cazul apariției unui astfel de semnal se transmite către interfață un semnal corespunzător; după transmiterea fiecărui cadru *driver*-ul de tratare a coliziunii din *transceiver* generează un semnal de test numit *CPT* (*Collision Presence Test*) cu scopul de a testa circuitul de sesizare a coliziunilor și de a informa interfața cu privire la corecta funcționare a acestuia;
- implementarea funcției de *jabber* (detectarea transmisiilor prea lungi datorate existenței unor cadre care depășesc lungimea maximă admisă);
- generarea semnalului de alimentare electrică pentru cablu și pentru circuitele electrice interne; la acest nivel se realizează separarea galvanică dintre transceiver și interfață.

Conectarea mecanică a *transceiver*-ului la cablul coaxial se realizează cu ajutorul unui conector de tip *vampir*. Acesta conține un șurub care perforează cablul în punctul dorit și atinge conductorul central al cablului coaxial. Astfel se realizează contactul necesar prelucrării semnalului din rețea.

Reguli de configurare

Pentru ca o rețea *Ethernet* să funcționeze în condiții optime trebuie respectate anumite reguli. Câteva dintre acestea sunt prezentate în continuare:

- lungimea maximă a unui segment coaxial nu poate să depășească 500 m;
- lungimea maximă a unui cablu *transceiver* nu poate să depășească 50 m;
- distanța minimă dintre două *transceiver*-e trebuie să fie de cel puțin 2,5 m;
- numărul maxim de stații care sunt legate la același segment trebuie să fie de cel mult 100;
- între oricare două stații nu pot exista mai mult de două repetoare; acesta se datorează faptului că repetoarele *Ethernet* nu regenerează integral preambulul folosit pentru sincronizare, anumite părți ale acestuia fiind eliminate (pierdute); astfel, la fiecare trecere printr-un repetor, lungimea preambulului se micșorează;
- este obligatorie prezența terminatorilor de rețea cu impedanța de 50Ω; aceștia au un rol foarte important în asigurarea reflexiei semnalelor.

Încheiere

Chiar dacă rețelele *Ethernet* au apărut cu destul de mult timp în urmă, implementarea lor este destul de simplă, motiv pentru care sunt încă folosite pe scară largă, chiar dacă viteza lor este mult mai mică decât a altor tipuri de rețele.

Un avantaj foarte important al rețelelor *Ethernet* îl reprezintă costurile foarte mici necesare realizării acestora. Chiar dacă viteza este de maximum 10 *Mbps*, în cazul în care acest factor nu are un rol decisiv, rețelele *Ethernet* rămân cea mai bună soluție pentru crearea rețelelor locale de calculatoare.

În ultimii ani au fost realizate diferite tipuri de rețele bazate pe standarde din aceeași familie, care funcționează la viteze mult mai mari, ajungându-se chiar și la valori de 1 *Gbps*.

În numărul următor

Prezentarea rețelelor de calculatoare va continua cu descrierea altor tipuri de rețele care au la bază standarde derivate din *IEEE* 802.3. În numărul 12/5 (mai 2002) al revistei noastre veți putea găsi detalii referitoare la standardele 10*Base*2, 10*Base*5, 10*Broad*36, *FOIRL*, 10*Base*T, 10*Base*FP, 10*Base*FB și 10*Base*FL.

Mihai Scorțaru este redactor-șef adjunct al GInfo. Poate fi contactat prin e-mail la adresa **skortzy@yahoo.com**.

Firme celebre...

MICROSOFT

Carmen Șușman

Compania Microsoft este una dintre cele mai cunoscute firme care își desfășoară activitatea în domeniul tehnologiei informației. În cadrul acestui articol veți putea citi câteva informații referitoare la istoria de peste un secol a acestei companii.

Nașterea companiei

În ianuarie 1975, compania *MITS* (*The Micro Instrumentation and Telemetry Systems*) a creat primul calculator personal, *Altair* 8800. Acest produs a apărut pe coperta revistei *Popular Electronics*, revistă care conținea un articol în care calculatorul a fost descris în detaliu. Articolul i-a fascinat pe ***Paul Allen*** și pe ***Bill Gates***, doi studenți, care au decis că este momentul să proiecteze *software* pentru astfel de calculatoare.

Cei doi prieteni au realizat, până în luna februarie, limbajul *BASIC* și l-au licențiat primului cumpărător: *MITS*. *BASIC* este primul limbaj de programare pentru *PC*-uri.

În luna martie, *Paul Allen* este angajat la *MITS* cu funcția de director de *software*.

În luna iulie, limbajul *BASIC* ajunge la versiunea 2.0 și este distribuit în două variante: prima de 4 KB și a doua de 8 KB.

Bill Gates a renunțat la facultate și, în luna noiembrie, i-a adresat o scrisoare lui *Paul Allen*, în care apăreau informații referitoare la viitoarea colaborare dintre ei și în care a folosit pentru prima dată numele *Microsoft*.

Istorie

În februarie 1976, *Bill Gates* a scris un articol în care acuza unele persoane că îi "fură" programul, ceea ce îl împiedica să realizeze un produs *software* performant.

În luna iulie, compania *Microsoft* a îmbunătățit limbajul *BASIC* și l-a vândut unor distribuitori ca *DTC*, *General Electric*, *NCR* și *Citibank*.

În luna noiembrie, *Paul Allen* a plecat de la *MITS* pentru a acorda mai mult timp companiei *Microsoft*. Tot în această lună denumirea *Microsoft* a fost înregistrată la *Biroul Secretariatului de Stat* din *New Mexico*.

În anul 1977, *Bill Gates* și *Paul Allen* au semnat un contract de colaborare. În luna iulie, *Microsoft* a pus în vânzare al doilea limbaj de programare pentru *PC*-uri, *FORTRAN*. La sfârșitul anului a fost cedată licența asupra limbajului *BASIC* companiei *MITS*, după ce s-au judecat îndelungat.

Pentru anul următor compania *Microsoft* avea planuri de a realiza un al treilea limbaj de programare, *Microsoft COBOL-80*, și a stabilit un birou de vânzări în *Tokyo*, *Japonia*. Rapoartele fiscale de la sfârșitul acestui an indicau o cifră de afaceri de peste un milion de dolari.

În anul 1979, compania și-a mutat sediul în *Bellvue*, *Washington*. În luna aprilie, produsul 8080 *BASIC* a câștigat premiul *ICP Million Dollar*.

În luna iulie, compania a anunțat o nouă variantă a limbajului *BASIC* destinată procesoarelor 8086 pe 16 biți. Aceasta a fost prima lansare a unui limbaj rezident de nivel înalt pentru calculatoare pe 16 biți.

În luna noiembrie, compania *Microsoft* s-a extins în *Europa*, având ca reprezentant pe *Vector Microsoft* în *Belgia*.

În aprilie 1980, *Microsoft* a lansat cardul Z-80 *SoftCard* (un microprocesor pe o placă cu circuite integrate), care se atașa calculatoarelor *Apple* II și permitea utilizatorilor să ruleze programe disponibile pentru calculatoarele din clasa 8080/Z-80 modificând foarte puțin sursele.

În luna iunie este angajat *Steve Ballmer*. Sarcina sa era de a realiza politicile de finanțare, organizare și alocare a resurselor companiei.

În august 1980, compania a anunțat produsul *XENIX OS*, un sistem de operare portabil pe majoritatea sistemelor echipate cu procesoare pe 16 biți. *XENIX* este un sistem de operare interactiv, *multi-user* și *muti-tasking*. Acesta era capabil să ruleze toate aplicațiile realizate de către *Microsoft* și era compatibil cu programe scrise pentru sistemul de operare *UNIX*.

În iunie 1981, *Microsoft* a devenit corporație privată avându-l pe *Bill Gates* ca președinte și conducător al adunării generale și pe *Paul Allen* ca vicepreședinte executiv.

În august 1981, compania *IBM* a lansat un calculator personal care utiliza un sistem de operare produs de *Microsoft* (*MS-DOS* 1.0) și limbaje de programare ca *BASIC*, *COBOL* și *PASCAL*.

În iulie 1982 a fost lansată rețeaua locală *MILAN* (*Microsoft Local Area Network*); era folosit sistemul de operare *XENIX*.

În mai 1983 a fost introdus un nou produs, *Microsoft Mouse*; dispozitivul era folosit pentru a deplasa cursorul pe ecran mai repede decât folosind metodele convenționale de atunci.

În luna septembrie *Microsoft* a lansat editorul de texte *Microsoft Word* pentru *MS-DOS* 1.0; toți abonații revistei *PC World* au primit o dischetă care conținea o versiune demonstrativă a acestui program.

Două luni mai târziu *Microsoft* a anunțat produsul *Microsoft Windows*, care urma să fie o suprafață de operare pentru *MS-DOS* și permitea utilizatorilor să ruleze simultan mai multe programe; era folosită o interfață grafică bazată pe ferestre.

În aprilie 1984 se creează un departament de componente *hardware* coordonat de *William Roland*. Rolul acestui departament era să dezvolte și să vândă componente *hardware* care să completeze linia de produse *software*.

În luna august compania anunță că va folosi sistemele de operare *XENIX* și *MS-DOS* pentru calculatoarele *IBM PC AT*. Noile calculatoare impun standardul pentru sistemele *multiuser*.

În august 1985 se sărbătoresc 10 ani de la crearea companiei și se prezice că până la sfârșitul anului aceasta va avea un profit de aproximativ 140 milioane de dolari.

În aprilie 1987 a fost lansat sistemul de operare *Microsoft Operating System/2 (MS OS/2)* și a fost anunțat produsul *Microsoft Windows* 2.0, care era compatibil cu versiunile anterioare și putea fi folosit pe platforma *OS/2*.

În luna iulie *Microsoft* a achiziționat *Forethought Inc.*, o companie care producea aplicații *software*. *Forethought* dezvoltă și lansează pe piață produsul *PowerPoint*, o aplicație destinată realizării de prezentări.

În luna septembrie *Microsoft* pune în vânzare prima aplicație stocată pe *CD*-uri, *Microsoft Bookshelf*.

În octombrie 1987 este lansat produsul *Microsoft Excel* pentru *Windows*, o aplicație destinată calculului tabelar.

În prima lună a anului 1988, *Microsoft* și *Ashton-Tate* anunță *SQL Server*, un server de baze de date relaționale pentru rețele locale.

În iunie 1989 *Microsoft* creează un nou departament care are ca scop realizarea de sisteme multimedia.

La sfârșitul anului *Microsoft* își prelungește contractul cu *IBM*; cele două companii sunt de acord să realizeze în deceniul următor produse *software* pentru toate categoriile de utilizatori.

În mai 1990 *Microsoft* anunță disponibilitatea pe piață a suprafeței de operare *Windows* 3.0. Noua versiune oferă o gamă largă de programe care aveau scopul de a facilita utilizarea calculatorului. În luna iulie, înainte de sărbătorirea a 15 ani de la nașterea companiei, vânzările depășesc un miliard de dolari. *Microsoft* devine prima companie producătoare de *software* cu venituri de peste un miliard de dolari într-un singur an fiscal.

În anul 1991 au fost lansate produsele *Excel* 3.0 pentru *Windows* 3.0, *Excel* pentru *Mac* și o versiune multimedia a produsului *Works* pentru *Windows* 2.0.

În aprilie 1992 *Microsoft* a pus în vânzare produsul *Windows* 3.1 care avea față de versiunea precedentă peste 1.000 de îmbunătățiri.

În luna iunie președintele *George Bush* i-a oferit lui *Bill Gates* o medalie pentru realizările sale tehnologice.

Spre sfârșitul anului au fost lansate *Microsoft Windows for Workgroups* 3.1 și *Microsoft Access*.

La începutul anului următor, la 10 ani de la lansarea procesorului de text *Microsoft Word*, statisticile arătau că acest produs era folosit de peste 10 millioane de utilizatori din întreaga lume.

În martie 1993 *Microsoft* a lansat *Encarta*, prima enciclopedie în format electronic. Aceasta conținea o mulțime de articole interesante, animații, sunete, grafice și fotografii realizate folosind o tehnologie de vârf.

La sfârșitul lunii a fost lansat un *upgrade* pentru sistemul de operare *MS-DOS* 6.0 care avea suport integrat pentru compresie de date (*DoubleSpace*).

În aprilie 1993 peste 25 de milioane de utilizatori dețineau licențe *Windows*.

La sfârșitul anului compania *Microsoft* a fost numită în paginile revistei *Fortune* "cea mai inovatoare companie".

În septembrie 1994 *Microsoft* și-a anunțat intenția de a lansa sistemul de operare *Windows* '95. Acesta a apărut pe piață în august 1995 și, până la sfârșitul lunii, s-au vândut peste un milion de licențe.

În cursul lunii noiembrie *Microsoft* a anunțat lansarea programului de navigare *Internet Explorer* 2.0 pentru *Windows* '95.

În ultimii ani compania *Microsoft* și-a diversificat foarte mult gama de produse. Au fost lansate o mulțime de sisteme de operare din familia *Windows*; câteva dintre ele sunt *Windows NT*, *Windows* '98, *Windows* 2000, *Windows Me* și *Windows XP*.

Principalele aplicații destinate utilizatorilor obișnuiți au fost grupate în pachetul *Microsoft Office*. Acesta conține produsele *Word*, *Excel*, *PowerPoint*, *Access* etc. De-a lungul timpului numărul aplicațiilor care fac parte din acest pachet a crescut. Cele mai cunoscute versiuni sunt *Office* '97, *Office* 2000 și *Office XP*.

Nici programatorii nu au fost neglijați de către *Microsoft*. A apărut o suită de medii de programare pentru diferite limbaje cum ar fi *Basic*, *C*, *Java*, *FoxPro* etc. Aceste medii au fost grupate în pachetul *Microsoft Visual Studio*.

O noutate interesantă este conceptul *Microsoft .NET* potrivit căruia, în viitor, toate dispozitivele electrice vor fi conectate într-o rețea globală. Primul produs din familia *.NET* este pachetul *Microsoft Visual Studio .NET*.

Microsoft s-a implicat și în producția de jocuri pentru *PC*-uri. Seria *Age of Empires*, una dintre cele mai cunoscute familii de jocuri de strategie, a fost produsă de această companie. Recent *Microsoft* a anunțat lansarea unei console pentru jocuri, denumită *XBox*.

Carmen Șușman este redactor GInfo și poate fi contactată prin e-mail la adresa **schummy@email.ro**.

Heroes of Might & Magic IV

Mihai Scorțaru

Pe data de 29 martie 2001 a fost lansat cel de-al patrulea joc din seria Heroes of Might and Magic. Acest articol este primul review complet al unui joc prezentat în Gazeta de Informatică. Veți putea afla o mulțime de informații despre unul dintre cele mai așteptate jocuri de strategie...

Heroes of Might and Magic este una dintre cele mai cunoscute serii de jocuri de strategie. Primele trei jocuri ale familiei s-au bucurat de un succes extraordinar, numărându-se printre cele mai apreciate jocuri în primele luni de după lansare. Conceptul care stă la baza jocurilor din această serie este foarte simplu: jucătorul trebuie să își conducă *eroii* spre victorie. Pentru aceasta el va trebui să dezvolte orașele pe care le stăpânește pentru a recruta diferite creaturi, pentru a ajuta la dezvoltarea eroilor și pentru a obține diferite resurse.

Cel mai important element al jocului îl constituie eroii. Aceștia sunt reprezentanții jucătorului în lumea magică a jocului. Eroii pot fi recrutați contra cost în tavernă, una dintre clădirile care pot fi construite în orașe.

Imediat după recrutare, eroul are la dispoziție o mică armată care îl va însoți în explorarea hărții. Pentru ca eroul să aibă șanse de câștig în luptele cu adversarii, el trebuie să aibă în subordine o armată cât mai puternică. Creaturile care vor face parte din armate trebuie recrutate fie în orașe, fie din diferite clădiri care apar pe hartă.

Pentru ca eroul să devină mai puternic, el trebuie să acumuleze cât mai multă experiență, să adune diferite obiecte magice (***artifacts*** - *artefacte*) sau să se antreneze.

Experiența poate fi câștigată în luptă sau efectuând diferite acțiuni. Artefactele pot fi găsite explorând harta sau pot fi capturate în urma luptelor cu adversarii. Antrenamentele pot avea loc în diferite clădiri de pe hartă sau în orașe.

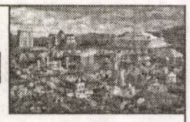
Un rol foarte important în desfășurarea jocului îl are ***magia***. Eroii au la dispoziție sau pot cumpăra o carte de magie, care le permite să folosească vrăji în timpul luptelor sau vrăji care să îi ajute în explorarea hărții. În această carte vor apărea diferite vrăji, pe măsură ce acestea sunt însușite. Cea mai importantă sursă de vrăji este ***ghilda magică***, o clădire care poate fi construită în orașe.

Pentru a putea recruta eroi sau creaturi și pentru a construi clădiri jucătorul are nevoie de diferite resurse. Tipul acestora nu a fost modificat de-a lungul celor patru jocuri ale seriei; există șapte tipuri de resurse și anume: ***aur*** (*gold*), ***pietre*** (*ore*), ***lemne*** (*wood*), ***sulf*** (*sulfur*), ***mercur*** (*mercury*), ***cristale*** (*crystals*) și ***pietre prețioase*** (*gem*).

Legendă...

At (Attack - Atac): caracterizează capacitățile ofensive;
Def (Defense - Apărare): caracterizează capacitățile defensive;
HP (Hit points - Puncte de viață): indică punctele de viață disponibile;
Dam (Damage - Daune): indică daunele cauzate adversarilor în urma unui atac;
SP (Spell points - Puncte de magie): indică punctele de viață disponibile;
Sh (Shots - Lovituri): indică numărul de atacuri la distanță într-o bătălie;
Speed (Speed - Viteză): indică ordinea în care vor intra în luptă creaturile;
MB (Movement in battle - Mișcare în luptă): numărul de poziții care pot fi traversate;
MB (Movement on map - Mișcare pe hartă): numărul de poziții care pot fi traversate;
Exp (Experience - Experiență): caracterizează experiența creaturii;
Cost (Cost - Cost): resursele necesare achiziționării (aur);
Pop (Population - Populație): numărul de indivizi care pot fi recrutați săptămânal;

Academy



Halfling - nivelul 1

At: 10	Def: 10	HP: 8	Dam: 1-2
Sh: 10	Speed: 6	MB: 4	MM: 18
Exp: 7	Cost: 22	Pop: 23	

Giant slayer: provoacă daune mai mari creaturilor de nivelul 4.

Dwarf - nivelul 1

At: 11	Def: 11	HP: 12	Dam: 2-3
Speed: 6	MB: 4	MM: 18	Exp: 8
Cost: 28	Pop: 29		

50% magic resistence: are 50% șanse să reziste la atacurile magice; daunele provocate de vrăji sunt reduse la 50%.

Mage - nivelul 2

At: 6	Def: 12	HP: 16	Dam: 3-4
SP: 16	Speed: 5	MB: 6	MM: 19
Exp: 39	Cost: 160	Pop: 8	

Spell caster: poate folosi anumite vrăji.

Gold golem - nivelul 2

At: 16	Def: 16	HP: 50	Dam: 8-12
Speed: 3	MB: 6	MM: 19	Exp: 31
Cost: 220	Pop: 6		

75% magic resistence: are 75% șanse să reziste la atacurile magice; daunele provocate de vrăji sunt reduse la 25%.

Mechanical: nu are moral și nu este influențat de vrăjile care afectează creaturile vii.

Genie - nivelul 3

At: 10	Def: 18	HP: 60	Dam: 9-12
SP: 24	Speed: 6	MB: 10	MM: 25
Exp: 105	Cost: 550	Pop: 6	

Flying: poate zbura.

Spell caster: poate folosi anumite vrăji.

Naga - nivelul 3

At: 22	Def: 22	HP: 90	Dam: 14-22
Speed: 7	MB: 6	MM: 19	Exp: 157
Cost: 850	Pop: 4		

No retaliation: adversarul nu poate răspunde atacului.

Titan - nivelul 4

At: 32	Def: 32	HP: 210	Dam: 34-50
Sh: 16	Speed: 6	MB: 7	MM: 21
Exp: 616	Cost: 4000 + 2Gems		Pop: 2

No melee penalty: daunele provocate adversarilor nu sunt reduse în cazul atacului "de aproape".

Chaos ward: este imun la vrăjile școlii haosului și sunt reduse daunele produse de creaturile din asylum.

Dragon golem - nivelul 4

At: 34	Def: 34	HP: 220	Dam: 36-55
Speed: 9	MB: 12	MM: 27	Exp: 630
Cost: 4000 + 4Ore		Pop: 2	

First strike: atacă înaintea inamicului.

Negate first strike: anulează efectul caracteristicii first strike a inamicilor.

Mechanical: nu are moral și nu este influențat de vrăjile care afectează creaturile vii.

În cazul în care doi eroi inamici se întâlnesc va începe lupta dintre ei. Va apărea ***ecranul de luptă*** (*combat screen*) în care jucătorul va putea controla modul în care se vor comporta eroul și creaturile în luptă.

Există multe alte aspecte interesante ale jocurilor din seria *Heroes of Might and Magic*, dar nu le vom prezenta detaliat în cadrul acestui articol, ci ne vom concentra asupra noutăților introduse de ultimul joc al seriei.

Prezentare generală

În *Heroes of Might and Magic* IV există șase tipuri de orașe: ***academy***, ***necropolis***, ***asylum***, ***haven***, ***preserve*** și ***stronghold***.

Fiecare dintre primele cinci tipuri de orașe este guvernat de unul dintre cele cinci tipuri de magie și îi sunt caracteristici eroi din două clase. Celui de-al șaselea oraș nu îi corespunde nici o școală de magie și îi este specific doar un singur tip de erou.

În *academy* pot fi recrutați ***lorzi*** (*lords*) și ***magicieni*** (*enchanters*), iar orașul este guvernat de magia ***ordinii*** (*order*). *Necropolis*-ul este orașul în care veți putea recruta ***războinici*** (*warriors*) și ***necromanți*** (*necromancers*) și veți avea acces la magia ***morții*** (*death*). *Asylum*-ul este guvernat de magia ***haosului*** (*chaos*) și permite recrutarea ***tâlharilor*** (*rogues*) și a ***vrăjitorilor*** (*sorcerers*). În *haven* veți putea beneficia de serviciile ***luptătorilor*** (*fighters*) și ale ***preoților*** (*priests*) și veți folosi mai ales magia ***vieții*** (*life*). *Preserve* este orașul corespunzător magiei ***naturii*** (*nature*) și în care veți putea recruta ***arcași*** (*archers*) și ***șamani*** (*shamans*). Așa cum am spus, ultimul tip de oraș, *stronghold*, permite recrutarea unui singur tip de eroi: ***barbari*** (*barbarians*) și nu se află sub incidența nici uneia dintre cele cinci școli de magie.

Eroii din cele 11 clase pot evolua și, pe măsură ce calitățile lor se îmbunătățesc, pot ajunge să facă parte din una dintre cele 37 de clase avansate.

Pe măsură ce acumulează experiență, eroii își pot îmbunătăți una dintre cele nouă calități primare: ***tactică*** (*tactics*), ***luptă*** (*combat*), ***noblețe*** (*nobility*), ***cercetare*** (*scouting*), ***magia vieții*** (*life magic*), ***magia ordinii*** (*order magic*), ***magia naturii*** (*nature magic*), ***magia haosului*** (*chaos magic*) și ***magia morții*** (*death magic*). Doar cinci dintre cele nouă calități pot fi însușite de către un erou și fiecare dintre ele permite dezvoltarea a câte trei calități secundare.

Armatele eroilor pot fi formate din 66 de tipuri de creaturi. Fiecărui tip de oraș îi corespund opt creaturi, iar celelalte pot fi recrutate prin diferite alte metode.

Probabil, cea mai importantă noutate a jocului este faptul că eroii au acum un rol activ în timpul luptelor. În primele trei jocuri ale seriei, eroii nu puteau participa efectiv la luptă, ci doar își puteau ajuta creaturile cu ajutorul vrăjilor.

Caracteristici

Surpinzător, noul joc nu necesită resurse *hardware* foarte mari. Cerințele minime sunt un procesor cu frecvența de

Asylum

Bandit - nivelul 1

At: 9	Def: 9	HP: 10	Dam: 1-3
Speed: 7	MB: 7	MM: 21	Exp: 7
Cost: 24	Pop: 23		

Stealth: invizibil în anumite situații.

Orc - nivelul 1

At: 11	Def: 9	HP: 12	Dam: 1-3
Sh: 6	Speed: 4	MB: 6	MM: 19
Exp: 10	Cost: 34	Pop: 16	

Ranged: poate ataca la distanță.

Short range: provoacă daune mai mari dacă ținta este mai apropiată.

No melee penalty: daunele provocate adversarilor nu sunt reduse în cazul atacului "de aproape".

Medusa - nivelul 2

At: 19	Def: 17	HP: 24	Dam: 3-6
Sh: n/a	Speed: 5	MB: 6	MM: 19
Exp: 52	Cost: 220	Pop: 6	

Ranged: poate ataca la distanță.

No melee penalty: daunele provocate adversarilor nu sunt reduse în cazul atacului "de aproape".

Unlimited shots: nu rămâne niciodată fără muniție.

Stone gaze: poate împietri adversarii (cu excepția celor din categoriile undead, mechanical și elemental).

Minotaur - nivelul 2

At: 16	Def: 15	HP: 38	Dam: 5-10
Speed: 6	MB: 6	MM: 19	Exp: 53
Cost: 230	Pop: 6		

Block: are 30% șanse să blocheze atacurile inamice fără a fi afectat.

Efreet - nivelul 3

At: 25	Def: 23	HP: 80	Dam: 10-20
Speed: 6	MB: 13	MM: 29	Exp: 161
Cost: 850	Pop: 4		

Flying: poate zbura.

Fire shield: provoacă daune suplimentare datorită scutului de foc.

Fire attack: atacă folosind foc; daunele provocate inamicilor cu rezistență la foc sunt mai mici.

Fire resistence: are rezistență la atacurile cu foc și este imun la vrăjile de foc.

Nightmare - nivelul 3

At: 23	Def: 20	HP: 110	Dam: 12-26
SP: 6	Speed: 6	MB: 9	MM: 23
Exp: 148	Cost: 800	Pop: 4	

Terror: adversarul poate pierde dreptul la mutare.

Life ward: este imun la vrăjile școlii vieții și sunt reduse daunele produse de creaturile din haven.

Hydra - nivelul 4

At: 30	Def: 26	HP: 250	Dam: 28-66
Speed: 6	MB: 7	MM: 21	Exp: 617
Cost: 4000 + 2 Mercury			Pop: 2

Multiple attack: atacă toți inamicii din pozițiile adiacente.

No retaliation: adversarul nu poate răspunde atacului.

Black Dragon - nivelul 4

At: 40	Def: 40	HP: 400	Dam: 55-110
Speed: 7	MB: 15	MM: 30	Exp: 1233
Cost: 8000 + 4 Sulfur		Pop: 1	

Flying: poate zbura.

Breath attack: poate ataca mai mulți adversari aflați unul în spatele altuia.

Magic immunity: nu este afectat de vrăji.

200 MHz, o memorie de 32 MB și 650 MB spațiu liber pe disc. Totuși, este recomandabil ca frecvența procesorului să fie de cel puțin 400 MHz și să aveți la dispoziție cel puțin 64 MB de memorie. Jocul nu necesită o placă video cu accelerator 3D. Rezoluția maximă care poate fi folosită este de 1280 × 1024 pixeli.

Resurse

Cea mai importantă modificare în ceea ce privește resursele apare în cazul minelor. Până acum, dacă un jucător reușea să intre în posesia unei mine, atunci el primea, în fiecare zi a jocului, resursele corespunzătoare. Acum, o mină se poate afla în diferite stadii de dezvoltare: ***depozit de resurse*** (*resource depozit*), ***mină în exploatare*** (*working mine*), ***mină abandonată*** (*abandoned mine*) și ***mină cu resurse epuizate*** (*run-out mine*). Pentru a beneficia de resursele din depozit, jucătorul va trebui să pună mina în exploatare contra cost. Minele abandonate pot fi folosite din nou dacă se câștigă lupta cu creaturile care le apără. Resurse mai pot fi obținute pe diferite alte căi, cum ar fi vizitarea unor clădiri.

Eroii

Pe lângă faptul că eroii au acum un rol activ în luptă, există o mulțime de alte modificări aduse în acest domeniu. Evident, aceștia pot fi doborâți în luptă; ei intră într-o stare de inconștiență și pot fi vindecați dacă restul armatei reușește să câștige bătălia. Dacă bătălia este pierdută, eroul poate ajunge în una dintre închisorile inamicului și, eventual, poate fi eliberat ulterior.

În timpul unei lupte, dacă este rândul eroului pentru a efectua o acțiune, jucătorul poate opta pentru un atac fizic, pentru folosirea unei vrăji sau a unei poțiuni magice.

Eroii pot călători acum și singuri, ei nemaiavând nevoie de creaturi care să îi însoțească. Mai mult, este posibil ca doi sau mai mulți eroi să călătorească împreună. De ase-

Haven

Squire - nivelul 1

At: 9 Def: 11 HP: 10 Dam: 2-3
Speed: 5 MB: 6 MM: 19 Exp: 7
Cost: 26 Pop: 21

Stunning: există șanse ca adversarul să își piardă rândul la mutare și să nu poată răspunde atacului.

Crossbowman - nivelul 1

At: 11 Def: 11 HP: 10 Dam: 2-3
Sh: 10 Speed: 2 MB: 6 MM: 19
Exp: 9 Cost: 32 Pop: 16

Ranged: poate ataca la distanță.

Long range: daunele nu sunt mai mici dacă ținta este mai depărtată.

Pikeman - nivelul 2

At: 12 Def: 14 HP: 38 Dam: 6-8
Speed: 4 MB: 6 MM: 19 Exp: 35
Cost: 150 Pop: 9

Long weapon: poate ataca inamicii din apropiere chiar dacă aceștia nu se află în poziții adiacente.

Negate first strike: anulează efectul caracteristicii first strike a inamicilor.

Ballista - nivelul 2

At: 16 Def: 15 HP: 38 Dam: 5-10
Speed: 6 MB: 6 MM: 19 Exp: 53
Cost: 230 Pop: 6

Ranged: poate ataca la distanță.

Long range: daunele nu sunt mai mici dacă ținta este mai depărtată.

No obstacle penalty: daunele nu sunt mai mici dacă ținta se află în spatele unui obstacol sau al unui zid.

Mechanical: nu are moral și nu este influențat de vrăjile care afectează creaturile vii.

Monk - nivelul 3

At: 20 Def: 22 HP: 55 Dam: 10-15
Speed: 5 MB: 6 MM: 19 Exp: 107
Cost: 550 Pop: 6

Ranged: poate ataca la distanță.

Death ward: este imun la vrăjile școlii morții și sunt reduse daunele produse de creaturile din necropolis.

Crusader - nivelul 3

At: 22 Def: 23 HP: 65 Dam: 10-16
Speed: 7 MB: 6 MM: 19 Exp: 124
Cost: 650 Pop: 5

Attack twice: atacă de două ori.

Death ward: este imun la vrăjile școlii morții și sunt reduse daunele produse de creaturile din necropolis.

Champion - nivelul 4

At: 25 Def: 25 HP: 150 Dam: 24-36
Speed: 6 MB: 10 MM: 25 Exp: 320
Cost: 2000 + 2Wood + 2Ore Pop: 4

First strike: atacă înaintea inamicului.

Charge: daunele provocate adversarului cresc proporțional cu distanța parcursă pentru lansarea atacului.

Angel - nivelul 4

At: 30 Def: 34 HP: 230 Dam: 40-65
SP: 20 Speed: 10 MB: 15 MM: 30
Exp: 620 Cost: 400 + 2Gem Pop: 2

Flying: poate zbura.

Ressurection: poate învia creaturi aliate.

menea, creaturile pot și ele călători singure, nici ele neavând nevoie de eroi care să le însoțească.

Sistemul de statistici al eroilor a fost modificat foarte mult. Principalele caracteristici ale eroilor sunt acum ***damage***, ***hit points***, ***shots***, ***luck***, ***morale***, ***speed***, ***movement*** și ***experience***.

Așa cum am spus, există nouă calități primare și 27 de calități secundare. Fiecărei calități primare îi corespunde un grup de trei calități secundare. Există cinci niveluri pentru cele 36 de calități și anume ***de bază*** (*basic*), ***avansat*** (*advanced*), ***expert*** (*expert*), ***maestru*** (*master*) și ***mare maestru*** (*grandmaster*). În continuare vă prezentăm calitățile principale de care depind calitățile secundare:

- *tactics*: ***offence***, ***defense*** și ***leadership***;
- *combat*: ***toughness***, ***archery*** și ***magic resistence***;
- *scouting*: ***pathfinding***, ***seamanship*** și ***stealth***;
- *nobility*: ***estates***, ***mining*** și ***diplomacy***;
- *life magic*: ***healing***, ***spirituality*** și ***resurrection***;
- *order magic*: ***enchantment***, ***wizardry*** și ***charm***;
- *death magic*: ***occultism***, ***demonology*** și ***necromancy***;
- *chaos magic*: ***conjuration***, ***pyromancy*** și ***sorcery***;
- *nature magic*: ***herbalism***, ***meditation*** și ***summoning***.

În momentul în care un erou obține caracteristicile necesare pentru a face parte dintr-o clasă avansată, el va deveni membru al acestei clase. În continuare vă prezentăm caracteristicile pe care le dețin eroii care fac parte din cele 48 de clase:

Erou	Caracteristici	Erou	Caracteristici
Death Night	tactics	Fire Diviner	scouting, chaos
Knight	tactics	Bard	scouting, nature
General	tactics, combat	Lord	nobility
Field Marchal	tactics, scouting	Cardinal	nobility, life
Lord Commander	tactics, nobility	Wizard King	nobility, order
Crusader	tactics, life	Dark Lord	nobility, death
Illusionist	tactics, order	Witch King	nobility, chaos
Reaver	tactics, death	Beast King	nobility, nature
Pyromancer	tactics, chaos	Priest	life
Warden	tactics, nature	Monk	life, order
Barbarian	combat	Dark Priest	life, death
Archer	combat	Heretic	life, chaos
Ranger	combat, scouting	Summoner	life, nature
Warlord	combat, nobility	Mage	order
Paladin	combat, life	Shadow Mage	order, death
Battle Mage	combat, order	Wizard	order, chaos
Assassin	combat, death	Enchanter	order, nature
Fireguard	combat, chaos	Necromancer	death
Beastmaster	combat, nature	Lich	death, chaos
Thief	scouting	Demonologist	death, nature
Guildmaster	scouting, nobility	Sorcerer	chaos
Prophet	scouting, life	Warlock	chaos, nature
Seer	scouting, order	Druid	nature
Ninja	scouting, death	ArchMage	trei școli de magie

Necropolis

Imp - nivelul 1
At: 10 Def: 10 HP: 7 Dam: 1-2
Speed: 6 MB: 12 MM: 28 Exp: 5
Cost: 18 Pop: 28

Flying: poate zbura.

Drain mana: transferă mana de la magii inamici la magii aliați.

Skeleton - nivelul 1

At: 12 Def: 12 HP: 8 Dam: 1-2
Speed: 4 MB: 6 MM: 19 Exp: 5
Cost: 22 Pop: 25

Resistence to ranged attacks: daunele provocate de atacurile de la distanță sunt înjumătățite.

Undead: nu are moral, este imun la anumite vrăji, dar este vulnerabil la vrăji speciale.

Ghost - nivelul 2
At: 15 Def: 30 HP: 15 Dam: 2-4
Speed: 6 MB: 12 MM: 28 Exp: 33
Cost: 140 Pop: 9

Flying: poate zbura.

Aging: reduce calitățile defensive ale adversarilor și daunele provocate de aceștia.

Undead: nu are moral, este imun la anumite vrăji, dar este vulnerabil la vrăji speciale.

Resistance to weapons: calități defensive ridicate împotriva atacurilor fizice.

Cerberus - nivelul 2

At: 16 Def: 16 HP: 24 Dam: 4-6
Speed: 6 MB: 7 MM: 21 Exp: 44
Cost: 190 Pop: 7

3-headed attack: poate ataca trei inamici aflați în apropiere.

No retaliation: adversarul nu poate răspunde atacului.

Vampire - nivelul 3
At: 30 Def: 30 HP: 75 Dam: 12-18
Speed: 8 MB: 12 MM: 28 Exp: 205
Cost: 1110 Pop: 3

Flying: poate zbura.

No retaliation: adversarul nu poate răspunde atacului.

Undead: nu are moral, este imun la anumite vrăji, dar este vulnerabil la vrăji speciale.

Drain life: folosește punctele de viață ale adversarilor pentru a-și învia trupele aliate.

Venom spawn - nivelul 3

At: 26 Def: 26 HP: 100 Dam: 16-24
Sh: 15 Speed: 6 MB: 6 MM: 19
Exp: 201 Cost: 1100 Pop: 3

Ranged: poate ataca la distanță.

Poisoning: otrăvește adversarii; aceștia pierd puncte de viață până la sfârșitul bătăliei.

Bone dragon - nivelul 4
At: 30 Def: 30 HP: 275 Dam: 45-65
Speed: 5 MB: 12 MM: 28 Exp: 621
Cost: 2000 + 2 Crystal Pop: 2

Flying: poate zbura.

Undead: nu are moral, este imun la anumite vrăji, dar este vulnerabil la vrăji speciale.

Fear: inamicii pot îngheța (nu răspund la atac) și se retrag câteva poziții.

Resistant to shots: daunele provocate de atacurile de la distanță sunt înjumătățite.

Devil - nivelul 4

At: 33 Def: 33 HP: 210 Dam: 34-50
SP: 8 Speed: 7 MB: 23 MM: 30
Exp: 614 Cost: 4000 + 2 Sulfur Pop: 2

Teleport: se poate teleporta în orice poziție în timpul luptei.

Summon demon: poate crea demoni de gheață (ice demons).

Life ward: este imun la vrăjile școlii vieții și sunt reduse daunele produse de creaturile din haven.

Creaturile

În noul joc își fac apariția 66 de creaturi, fiecare dintre ele având câte o caracteristică specială.

O noutate interesantă este faptul că acestea pot călători fără a fi însoțite de eroi. Ele pot culege resurse sau artefacte, dar doar eroii pot beneficia de avantajele aduse de obiectele magice. Totuși, creaturile pot folosi poțiuni.

Creaturile neutre care apar pe hartă se pot mișca acum și, în cazul în care consideră că pot câștiga o anumită bătălie, ele pot ataca. Există chiar și eroi neutri care se mișcă pe hartă la fel ca eroii jucătorilor, dar care nu culeg obiecte sau resurse.

Un nou concept interesant este cel de ***caravană***. Până acum, dacă se dorea ca un grup de creaturi să fie mutate dintr-un oraș în altul, atunci un erou trebuia să le transporte. Acum poate fi folosită o caravană pentru realizarea transportului: creaturile vor dispărea de pe hartă și vor apărea la destinație după intervalul de timp care ar fi fost necesar pentru realizarea transportului.

În fiecare tip de oraș pot fi construite clădiri care permit recrutarea anumitor creaturi. Pentru creaturile de nivel superior nu poate fi construită decât una dintre cele două clădiri. Astfel, va trebui să alegeți tipul creaturilor de nivel doi, trei și patru pe care doriți să le aveți la dispoziție. Ambele creaturi de nivel unu pot fi recrutate în oricare dintre orașe.

În jocurile anterioare noi creaturi deveneau disponibile pentru recrutare la începutul fiecărei săptămâni. În *He-*

Preserve

Sprite - nivelul 1
At: 9 Def: 10 HP: 7 Dam: 1-2
Speed: 6 MB: 13 MM: 29 Exp: 7
Cost: 22 Pop: 23

Flying: *poate zbura.*

No retaliation: *adversarul nu poate răspunde atacului.*

Wolf - nivelul 1
At: 9 Def: 9 HP: 12 Dam: 2-3
Speed: 7 MB: 7 MM: 21 Exp: 11
Cost: 36 Pop: 15

Attack twice: *atacă de două ori.*

Leprechaun - nivelul 1
At: 10 Def: 10 HP: 8 Dam: 1-2
SP: 3 Speed: 6 MB: 6 MM: 19
Exp: 7 Cost: 24 Pop: 22

Fortune: *poate folosi vraja norocului (fortune).*

Elf - nivelul 2
At: 16 Def: 16 HP: 18 Dam: 3-4
Sh: 16 Speed: 6 MB: 6 MM: 19
Exp: 43 Cost: 190 Pop: 7

Ranged: *poate ataca la distanță.*

Ranged first strike: *trage primul cu arcul.*

Shoots twice: *trage cu arcul de două ori.*

White tiger - nivelul 2
At: 19 Def: 19 HP: 35 Dam: 6-8
Speed: 8 MB: 9 MM: 24 Exp: 53
Cost: 220 Pop: 6

First strike: *atacă înaintea inamicului.*

Satyr - nivelul 2
At: 14 Def: 13 HP: 36 Dam: 5-8
SP: 6 Speed: 6 MB: 6 MM: 20
Exp: 39 Cost: 170 Pop: 8

Mirth: *poate folosi vraja veseliei (mirth).*

Griffin - nivelul 3
At: 19 Def: 19 HP: 95 Dam: 16-22
Speed: 6 MB: 13 MM: 29 Exp: 154
Cost: 800 Pop: 4

Flying: *poate zbura.*

Unlimited retaliation: *răspunde la toate atacurile.*

Unicorn - nivelul 3
At: 20 Def: 20 HP: 80 Dam: 12-20
Speed: 6 MB: 10 MM: 25 Exp: 120
Cost: 650 Pop: 5

Blind: *își poate orbi adversarii.*

Waswort - nivelul 3
At: 22 Def: 22 HP: 60 Dam: 10-14
Sh: 20 Speed: 4 MB: 1 MM: 13
Exp: 105 Cost: 550 Pop: 6

Ranged: *poate ataca la distanță.*

Weakness: *poate "slăbi" adversarul.*

Phoenix - nivelul 4
At: 28 Def: 28 HP: 275 Dam: 45-65
SP: 12 Speed: 12 MB: 15 MM: 30
Exp: 609 Cost: 4000 + 2 Mercury Pop: 3

Flying: *poate zbura.*

Rebirth: *poate renaște o singură dată în timpul unei bătălii.*

Breath attack: *poate ataca mai mulți adversari aflați unul în spatele altuia.*

Fire resistence: *are rezistență la atacurile cu foc și este imun la vrăjile de foc.*

Faerie dragon - nivelul 4
At: 15 Def: 29 HP: 220 Dam: 34-50
SP: 40 Speed: 7 MB: 13 MM: 29
Exp: 612 Cost: 4000 + 2 Gem Pop: 2

Flying: *poate zbura.*

Spell caster: *poate folosi anumite vrăji.*

Magic mirror: *există șanse ca vrăjile ofensive lansate asupra sa să se întoarcă împotriva inamicului.*

Mantis - nivelul 4
At: 34 Def: 34 HP: 210 Dam: 34-50
Speed: 8 MB: 13 MM: 29 Exp: 614
Cost: 4000 + 2 Gem Pop: 2

Flying: *poate zbura.*

First strike: *atacă înaintea inamicului.*

Binding: *inamicii vor ataca doar cu jumătate din forță.*

roes of Might and Magic IV această restricție a fost eliminată; în orașe apar noi recruți mult mai des, în funcție de rata de creștere a populației. Astfel, dacă rata de creștere este 7, va apărea câte un recrut în fiecare zi; dacă valoarea este 14, vor apărea câte doi recruți pe zi; pentru valori cuprinse între 8 și 13 vor apărea zilnic, unul sau doi recruți (dacă valoarea ar fi 10 atunci, în total, vor fi recrutate 10 creaturi în fiecare săptămînă). Rata de creștere depinde, în principal, de tipul creaturii (fiecare are o rată de creștere de bază). Aceasta poate fi mărită prin diferite procedee; unul dintre acestea este desemnarea unui guvernator al orașului (unul dintre eroi) - în funcție de noblețea guvernatorului, rata de creștere a tuturor creaturilor din oraș se va mări. Fiecare creatură are cel puțin o abilitate specială care îi permite să efectueze anumite acțiuni în timpul luptei sau aduce anumite beneficii posesorului.

Elementali

(Preserve)

Air elemental - nivelul 3

At: 16	Def: 32	HP: 40	Dam: 6-10
Speed: 7	MB: 13	MM: 29	Exp: 74
Cost: 350	Pop: 6		

Flying: poate zbura.

Resistance to weapons: calități defensive ridicate împotriva atacurilor fizice.

Elemental: imun la vrăjile care afectează creaturile vii.

Earth elemental - nivelul 3

At: 18	Def: 20	HP: 50	Dam: 9-14
Speed: 1	MB: 4	MM: 18	Exp: 77
Cost: 375	Pop: 6		

50% magic resistence: are 50% șanse să reziste la atacurile magice; daunele provocate de vrăji sunt reduse la 50%.

Elemental: imun la vrăjile care afectează creaturile vii.

Fire elemental - nivelul 3

At: 20	Def: 18	HP: 50	Dam: 7-10
Sh: 20	Speed: 6	MB: 6	MM: 20
Exp: 75	Cost: 350	Pop: 6	

Ranged: poate ataca la distanță.

Elemental: imun la vrăjile care afectează creaturile vii.

Fire attack: atacă folosind foc; daunele provocate inamicilor cu rezistență la foc sunt mai mici.

Fire resistence: are rezistență la atacurile cu foc și este imun la vrăjile de foc.

Water elemental - nivelul 3

At: 17	Def: 17	HP: 38	Dam: 6-9
Speed: 5	MB: 6	MM: 19	Exp: 73
Cost: 350	Pop: 6		

Spell caster: poate folosi anumite vrăji.

Elemental: imun la vrăjile care afectează creaturile vii.

Cold attack: atacă folosind gheață; daunele provocate inamicilor cu rezistență la gheață sunt mai mici.

Cold resistence: are rezistență la atacurile cu gheață și este imun la vrăjile de gheață.

Magia

Fiecare *școală de magie* corespunde unui anumit tip de oraș. Cele cinci școli de magie sunt foarte bine individualizate, diferențele dintre ele fiind foarte mari.

Cu toate acestea, fiecare aduce diverse beneficii și, în ansamblu, sistemul de magie este foarte bine echilibrat; nici una dintre cele cinci clase de magie nu aduce avantaje decisive.

Fiecare dintre școlile de magie are două școli aliate și două opuse.

Dacă privim sistemul de magie ca fiind o stea cu cinci colțuri, atunci fiecare școală de magie va fi aliată cu școlile din colțurile învecinate ale stelei și opusă celorlalte două școli.

Un nou element introdus în *Heroes of Might and Magic* IV este reprezentat de ***poțiunile magice***. Acestea au efecte similare vrăjilor de niveluri inferioare și pot fi găsite în diferite locuri sau pot fi cumpărate. Ele pot fi folosite și de către creaturile care nu sunt însoțite de eroi.

Vrăjile sunt împărțite pe cinci niveluri. Pentru a putea folosi vrăjile unei anumite școli, un erou trebuie să aibă caracteristica ce permite învățarea lor. Cu cât nivelul caracteristicii este mai mare, cu atât crește nivelul vrăjilor care pot fi învățate. Astfel, pentru nivelul de bază vor putea fi învățate doar vrăji de nivelul unu; un erou avansat va putea folosi și vrăji de nivelul doi; un expert va avea acces la vrăjile de nivelul trei, iar un maestru va putea beneficia și de vrăjile de nivelul patru; doar marii maeștri vor putea folosi vrăjile de nivelul cinci.

Într-un oraș va putea fi construit un turn de magie (denumirea clădirii diferă în funcție de tipul orașului) care va permite învățarea de vrăji ale școlii corespunzătoare orașului. Și această clădire are mai multe niveluri, construirea fiecăruia ducând la obținerea accesului la vrăjile de un anumit nivel. De asemenea, pot fi construite două librării (denumirea diferă în funcție de oraș), care permit accesul la vrăjile școlilor aliate. Această clădire nu are niveluri, nivelurile vrăjilor fiind determinate de nivelul turnului de magie.

Vă prezentăm în continuare efectele câtorva vrăji de nivelul cinci:

Vrajă	Școală	Descriere
Armageddon	haos	*aduce daune masive tuturor creaturilor de pe câmpul de luptă*
Chain Lighting	haos	*aduce daune masive câtorva dintre creaturile de pe câmpul de luptă*
Disintegrate	haos	*aduce daune masive unei creaturi; aceasta nu va mai putea fi înviată*
Dragon Strength	natură	*punctele de viață și daunele produse de o creatură se dublează*
Divine Intervention	viață	*efectul unei părți din daunele care au afectat trupele aliate este înlăturat*
Guardian Angel	viață	*o creatură primește un număr de vieți care depinde de calitățile magice ale eroului*
Hand of Death	moarte	*un număr de creaturi sunt eliminate instantaneu*
Hypnotize	ordine	*preia controlul asupra unităților inamice timp de trei tururi;*
Pain Mirror	ordine	*toate daunele care afectează o creatură aliată se reflectă și asupra adversarului*
Raise Vampire	moarte	*creează vampiri din cadavrele creaturilor*
Sacrifice	moarte	*sacrifică creaturi pentru a învia altele (numărul punctelor de viață ale creaturilor înviate este de două ori mai mare decât al creaturilor sacrificate)*
Steal All Enchantments	haos	*toate vrăjile benefice care afectează creaturile inamice sunt preluate și distribuite aleator între aliați*
Summon Devils	moarte	*creează un număr de devils*
Summon Fairy Dragon	natură	*creează un număr de fairy dragons*
Summon Phoenix	natură	*creează un număr de phoenix*
Summon Mantis	natură	*creează un mantis*

Stronghold

Berserker - nivelul 1

At: 12 Def: 8 HP: 18 Dam: 2-3
Speed: 6 MB: 6 MM: 20 Exp: 8
Cost: 20 Pop: 19

Attack twice: atacă de două ori.

Berserk: nu poate fi controlat; atacă din proprie inițiativă cel mai apropiat inamic.

Centaur - nivelul 1

At: 10 Def: 10 HP: 20 Dam: 3-5
Sh: 4 Speed: 4 MB: 10 MM: 25
Exp: 18 Cost: 40 Pop: 9

Ranged: poate ataca la distanță.

Short range: provoacă daune mai mari dacă ținta este mai apropiată.

No melee penalty: daunele provocate adversarilor nu sunt reduse în cazul atacului "de aproape".

Harpy - nivelul 2

At: 16 Def: 15 HP: 24 Dam: 4-5
Speed: 5 MB: 12 MM: 28 Exp: 39
Cost: 110 Pop: 8

Flying: poate zbura.

No retaliation: adversarul nu poate răspunde atacului.

Strike and return: după atac se întoarce în poziția din care a pornit.

Nomad - nivelul 2

At: 16 Def: 13 HP: 45 Dam: 6-9
Speed: 8 MB: 12 MM: 27 Exp: 52
Cost: 150 Pop: 6

First strike: atacă înaintea inamicului.

Ogre Mage - nivelul 3

At: 18 Def: 16 HP: 85 Dam: 12-18
SP: 6 Speed: 4 MB: 6 MM: 19
Exp: 101 Cost: 350 Pop: 6

Bloodlust: poate folosi vraja sângelui (bloodlust).

Cyclope - nivelul 3

At: 30 Def: 24 HP: 95 Dam: 12-18
Sh: 8 Speed: 4 MB: 7 MM: 21
Exp: 203 Cost: 750 Pop: 3

Ranged: poate ataca la distanță.

Area attack: toate trupele aflate în jurul țintei sunt afectate (chiar dacă sunt trupe aliate).

Thunderbird - nivelul 4

At: 30 Def: 30 HP: 170 Dam: 28-48
Speed: 6 MB: 13 MM: 29 Exp: 406
Cost: 1800 + 2 Mercury Pop: 3

Flying: poate zbura.

Lightning attack: după atac, folosește și vraja fulgerului (lightning) împotriva inamicului

Behemoth - nivelul 4

At: 36 Def: 34 HP: 240 Dam: 55-80
Speed: 6 MB: 7 MM: 21 Exp: 630
Cost: 2750 + 2 Crystal Pop: 2

Strength: provoacă daune mai mari.

Castelele

Fiecare dintre cele șase tipuri de castele permite construirea unor clădiri specifice care aduc diferite beneficii. În general, există opt tipuri de clădiri care permit recrutarea creaturilor (în *Preserve* mai apare una care permite recrutarea mai multor tipuri de creaturi), trei clădiri necesare pentru învățarea vrăjilor (în *Stronghold* lipsesc), o tavernă care permite recrutarea eroilor etc.

La fel ca în *Heroes of Might and Magic III*, pot fi construite mai multe turnuri și ziduri de apărare.

O noutate introdusă de *Heroes of Might and Magic IV* este faptul că nu mai pot fi construite toate clădirile. În funcție de "arborele de construcție" corespunzător fiecărui oraș, jucătorul va trebui să aleagă clădirile pe care dorește să le construiască.

Elemente noi au fost introduse și în cazul asediilor asupra orașelor. Până acum, infanteria nu putea ataca peste ziduri, ceea ce o făcea aproape inutilă. Acum, o creatură aflată de o parte a zidului poate ataca creaturile aflate de cealaltă parte.

Au fost introduse noutăți și în ceea ce privește arcașii din turnuri.

Jocul multiplayer

Cu siguranță acesta este unul dintre cele mai importante și mai interesante elemente ale unui joc. Din nefericire, în *Heroes of Might and Magic* IV, modul *multiplayer* ***lipsește cu desăvârșire.***

Dacă încercați să apăsați pe butonul ***Multiplayer*** sunteți rugat să așteptați primul *update* al jocului.

Campanii și scenarii

Pentru a vă bucura de acest joc aveți la dispoziție șase campanii, câte una pentru fiecare dintre tipurile de orașe.

De asemenea, producătorii au inclus 24 de scenarii în care vă veți confrunta, deocamdată, doar cu jucători controlați de calculator.

Vă recomadăm ca, mai întâi, să parcurgeți tutorialul chiar dacă ați jucat celelalte jocuri ale seriei. Există o mulțime de modificări față de acestea, *Heroes of Might and Magic* IV fiind un joc aproape în întregime nou.

De ce?

Heroes of Might and Magic IV se desfășoară într-o nouă lume, deoarece ținuturile în care a avut loc acțiunea celor-

Creaturi neutre

Pirate - nivelul 1 (Chaos)
At: 11 Def: 9 HP: 10 Dam: 1-2
Speed: 5 MB: 6 MM: 9 Exp: 7
Cost: 22 Pop: 24

Bonuses at sea: *pe mare are viteză, capacități ofensive și defensive mai mari și provoacă daune mai mari.*

Troglodyte - nivelul 1 (Chaos)
At: 11 Def: 9 HP: 14 Dam: 2-3
Speed: 5 MB: 6 MM: 19 Exp: 8
Cost: 28 Pop: 19

Immune to visual attacks: *vrăjile sau capacitățile adversarilor care afectează vederea nu au efect asupra sa.*

Zombie - nivelul 1 (Death)
At: 8 Def: 10 HP: 24 Dam: 2-4
Speed: 1 MB: 4 MM: 18 Exp: 10
Cost: 35 Pop: 15

Toughness: *număr mai mare de puncte de viață.*

Undead: *nu are moral, este imun la anumite vrăji, dar este vulnerabil la vrăji speciale.*

Peasant - nivelul 1 (Life)
At: 6 Def: 7 HP: 10 Dam: 2-3
Speed: 2 MB: 4 MM: 18 Exp: 4
Cost: 15 Pop: 36

Tax Payer: *aduce aur în visterie (1 gold / zi).*

Beholder - nivelul 2 (Chaos)
At: 16 Def: 15 HP: 26 Dam: 3-7
Sh: 15 Speed: 6 MB: 6 MM: 19
Exp: 51 Cost: 220 Pop: 6

Flying: *poate zbura.*

Ranged: *poate ataca la distanță.*

Random harmfull spell: *atacul este însoțit de o vrajă dăunătoare adversarului.*

Troll - nivelul 2 (Chaos)
At: 16 Def: 15 HP: 45 Dam: 6-12
Speed: 3 MB: 4 MM: 18 Exp: 54
Cost: 230 Pop: 6

Regeneration: *se vindecă la sfârșitul fiecărui tur al luptei.*

Gargoyle - nivelul 2 (Death)
At: 14 Def: 16 HP: 22 Dam: 4-6
Speed: 7 MB: 13 MM: 29 Exp: 38
Cost: 160 Pop: 8

Flying: *poate zbura.*

Elemental: *imun la vrăjile care afectează creaturile vii.*

Stone skin: *capacitățile defensive sunt cu 25% mai mari; nu poate beneficia și de vraja pietrificării pielii (stone skin).*

Mummy - nivelul 2 (Death)
At: 15 Def: 16 HP: 30 Dam: 5-8
Speed: 4 MB: 6 MM: 19 Exp: 39
Cost: 170 Pop: 8

Undead: *nu are moral, este imun la anumite vrăji, dar este vulnerabil la vrăji speciale.*

Curse: *atacurile sale sunt însoțite de vraja blestemului (curse).*

Mermaid - nivelul 2 (Might)
At: 15 Def: 13 HP: 38 Dam: 5-8
Speed: 5 MB: 6 MM: 19 Exp: 39
Cost: n/a Pop: 2

Hypnotize: *poate hipnotiza creaturile inamice; acestea vor lupta de partea sa timp de un tur.*

Ice demon - nivelul 3 (Death)
At: 30 Def: 30 HP: 70 Dam: 12-16
Speed: 5 MB: 6 MM: 19 Exp: 157
Cost: 850 Pop: 4

Freezing attack: *adversarul poate îngheța pentru două tururi.*

Cold attack: *atacă folosind gheață; daunele provocate inamicilor cu rezistență la gheață sunt mai mici.*

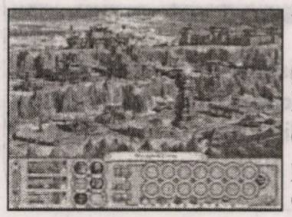
Cold resistence: *are rezistență la atacurile cu gheață și este imun la vrăjile de gheață.*

Sea monster - nivelul 4 (Might)
At: 35 Def: 34 HP: 275 Dam: 45-65
Speed: 5 MB: 9 MM: 24 Exp: 624
Cost: n/a Pop: 2

Devouring: *poate devora orice creatură în timpul atacului.*

alte trei jocuri ale seriei au fost distruse în urma unui cataclism.

Cauzele cataclismului sunt explicate în secvențele video de la începutul jocului. În *Heroes of Might and Magic* III: *Armageddon Blade* (primul *add-on* al jocului) scopul campaniei principale era obținerea artefactului *Armageddon Blade*. În *Heroes of Might and Magic* III: *Shadow of Death* (al doilea *add-on*) scopul unei campanii era obținerea artefactului *Sword of Frost*.



Suntem avertizați că atingerea celor două săbii va duce la declanșarea unui cataclism care va distruge lumea. Campania *Shadow of Death* se încheie prin cucerirea castelului în care era păstrat artefactul și cu constatarea faptului că acesta a fost furat.

La puțin timp după desfășurarea evenimentelor din cele trei jocuri *Heroes of Might and Magic* III, cei doi posesori ai artefacturilor s-au întâlnit în luptă. În momentul atingerii celor două săbii cataclismul s-a declanșat.

Din fericire, de fiecare dată există o portiță de salvare: imediat s-a deschis o poartă interdimensională spre o altă lume și o mulțime dintre locuitorii *Erathiei* au reușit să se salveze.

O nouă lume, un nou început, un nou prilej de aventură...

Mihai Scorțaru este redactor-șef adjunct al GInfo. Poate fi contactat prin e-mail la adresa **skortzy@yahoo.com**.

În numărul URMĂTOR

Ultimul număr dinaintea vacanței de vară al revistei noastre va apărea la sfârșitul lunii mai. Vă prezentăm pe scurt câteva dintre articolele pe care le veți putea citi...

ONI 2002

Cu siguranță, pentru cititorii care participă la concursurile de programare, unul dintre cele mai importante evenimente ale anului este Olimpiada Națională de Informatică. În acest an, concursul se va desfășura la Brăila în prima săptămână a vacanței de Paște.

În numărul următor veți putea citi câteva informații referitoare la modul în care s-a desfășurat ONI 2002; vor fi publicate enunțurile tuturor problemelor propuse spre rezolvare în cadrul concursului pe clase, precum și cele folosite pentru selecționarea componenților lotului național, dintre care vor fi aleși reprezentanții României la concursurile internaționale din acest an. De asemenea, vom publica listele premianților, precum și lista membrilor lotului național lărgit.

Bursele Agora

În numărul următor veți putea citi prezentarea rezolvărilor celor trei probleme ale ultimei runde a primei faze a concursului nostru. De asemenea, veți găsi clasamentul acestei ultime etape, clasamentul final al concursului, precum și câteva detalii referitoare la faza finală care se va desfășura "în timp real" la Liceul de Informatică din Cluj-Napoca și la care vor participa primii zece clasați în urma celor cinci etape ale primei faze a concursului.

Criptare

Serialul dedicat criptării va continua în următorul număr al Ginfo. Vă vom prezenta câteva metode performante care pot fi folosite pentru a îmbunătăți siguranța transmiterii sau păstrării informațiilor de importanță deosebită.

Concursuri

Înaintea olimpiadei naționale au loc diferite concursuri care au ca principal rol pregătirea elevilor pentru cel mai important concurs al anului. În acest număr ați putut citi enunțurile problemelor folosite la trei astfel de concursuri locale de programare.

În cadrul numărului 12/5 al Gazetei de Informatică vom continua prezentarea enunțurilor problemelor de la câteva concursuri de acest gen. Veți găsi în paginile revistei enunțurile problemelor propuse spre rezolvare la două astfel de competiții: Concursul interjudețean de programare "Ștefan Odobleja" care s-a desfășurat la Craiova și Concursul de informatică "Urmașii lui Moisil", desfășurat la Iași.

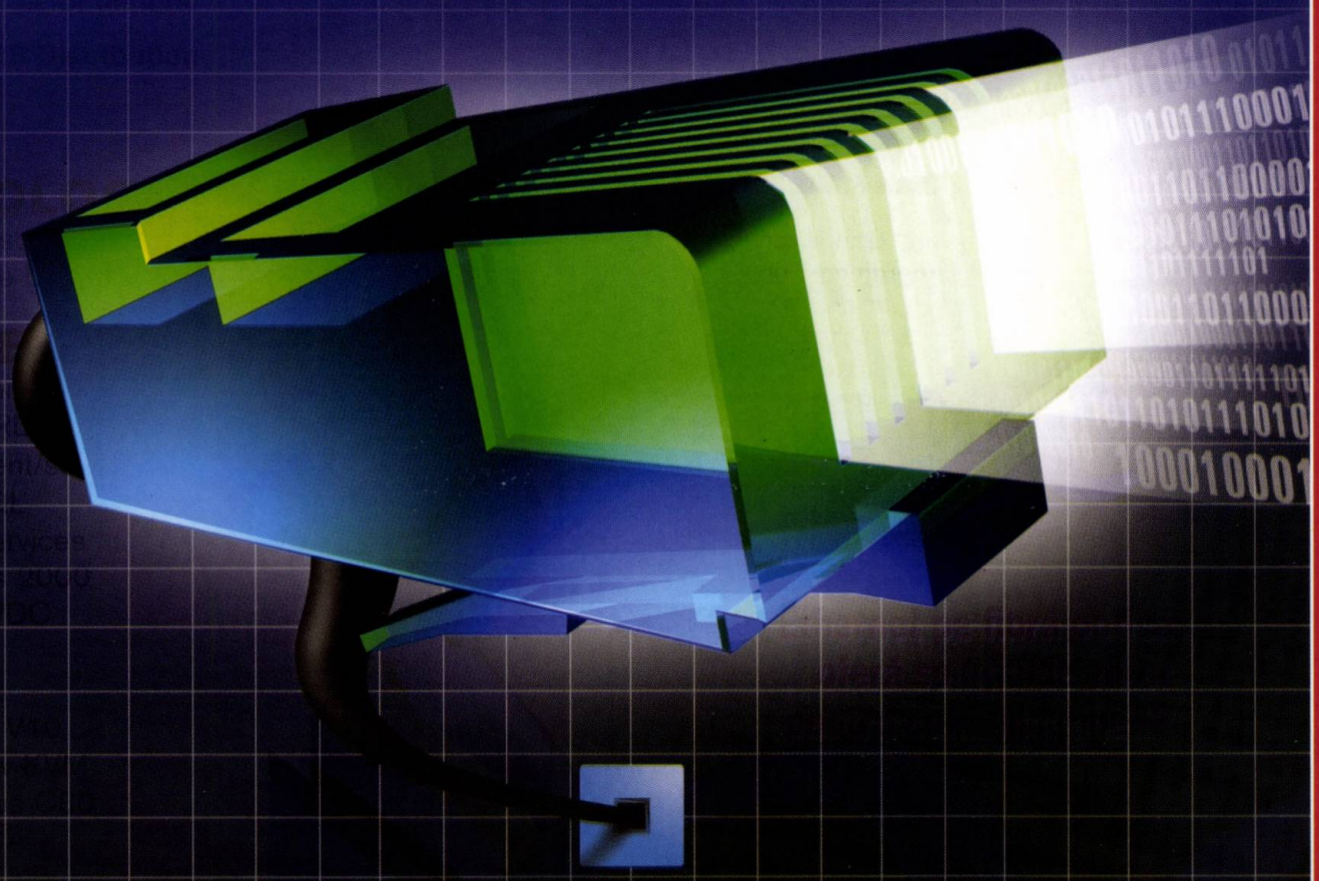
Rețele de calculatoare

Prezentarea rețelelor de calculatoare va continua cu detalii referitoare la diferite tipuri de astfel de rețele. Standardele pe care le vom prezenta sunt 10Broad36, FOIRL, 10Base2, 10Base5, 10BaseT, 10BaseFP, 10BaseFB, și 10BaseFL.

Sisteme nesecvențiale

Serialul dedicat sistemelor concurente, paralele și distribuite se va încheia cu prezentarea mecanismelor de comunicare între procese - (IPC - InterProcess Communication), instrumente puse la dispoziție de limbajele de programare pentru implementarea sistemelor nesecvențiale.

NETREPORT



Conectat la viitorul digital

AGORA MEDIA: Str. Tudor Vladimirescu nr. 63/9, Tg. Mureș, CP 230/1, 4300
Tel: 065-166 516, Fax: 065-166 290, e-mail: office@agora.ro

www.netreport.ro

Agora MEDIA

Publicarea acestei reviste este posibilă datorită sponsorizărilor acordate de următoarele firme care s-au decis să sprijine învățământul românesc de informatică

SPONSORI DE PLATINĂ

COMPAQ

SPONSORI DE AUR

FORTE COMPUTERS
Flexibility. Made in Romania

ASTRAL TELECOM

SPONSORI DE ARGINT

sat
IT SOLUTIONS & SERVICES

hp
invent

ERICSSON

Magua

M.G.T.
EDUCATIONAL

a&C
autodesk
authorized distributor

ALCATEL

*Rugăm companiile care doresc să se alăture acestui efort să ne contacteze prin e-mail la adresa office@ginfo.ro sau la numărul de telefon: 065-166.516

gazeta de informatica
Revista viitorilor specialiști în tehnologia informației
Mai 2002
Vol. 12/5
Lei 20.000

din sumarul numărului:

40 de probleme propuse spre rezolvare

- Olimpiada Națională de Informatică 2002
- Concursul de informatică "Ștefan Odobleja"
- Concursul de informatică "Urmașii lui Moisil"

Bursele Agora • Concurs: clasament final • ONI 2002
Tehnici de criptare • Tipuri de rețele • SuSE Linux 8.0
Instrumente de programare nesecvențiale

GHIDUL TĂU ÎN LUMEA TEHNOLOGIILOR

de reținut!

CONCURS DE FOTOGRAFIE DIGITALĂ

HP România –
PC Magazine România

1 martie - 24 mai 2002

www.hp.com.ro, www.pcmagazine.ro

Misiunea noastră este de a testa și evalua produse și servicii din domeniul tehnologiei informației și comunicațiilor și de a prezenta rezultatele cu onestitate și obiectivitate

AGORA MEDIA Str. C-tin Rădulescu Motru nr. 13, etaj 4, apt. 42, sect. 4 (CP 94/49), București
Tel: 01-3309282, Fax: 01-3309285

www.pcmagazine.ro

Agora MEDIA

Vol. 12, Nr. 5 /2002

gazeta de informatică

Redacția:

Director științific:
prof. dr. *Horia Georgescu*
Redactor șef:
lect. univ. *Clara Ionescu*
Redactor șef adjunct:
Mihai Scorțaru
Redactori:
prof. *Eugen Ionescu*
Carmen Șușman
Claudiu Soroiu
Au mai colaborat:
Emanuela Cerchez (Iași)
Oana Georgescu (Brașov)
Mihai Stroe (București)
Anca Vasilescu (Brașov)
Tehnoredactare:
Claudiu Soroiu
Coperta și grafica:
Andrei Pașa
Adresa pentru corespondență:
Computer Libris Agora,
Universității 8/8,
3400, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 064/192.422
065/166.516
Fax: 065/166.290
http://www.ginfo.ro
ISSN: 1222-7129

Tiparul: *Master Druck* Târgu Mureș
tel: 065/162.359

Editor: Agora Media SRL
Agora Media SRL, editor al publicațiilor "*eWeek România*", "*PC Magazine România*", "*NetReport*" și "*Gazeta de Informatică*", este membru al Biroului Român pentru Auditarea Tirajelor.
str. T. Vladimirescu nr. 63/9
4300 Târgu-Mureș
Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director tehnic:
Szabó László
Vânzare reclamă:
Mihai Bucuroiu
Bogdan Manu
Denis Marioara
Marius Comărniceanu
Reprezentanța București:
Director: *Dana Pastramă*
Tel: 01/330.92.81
Fax: 01/330.92.85

Sumar

actual.. PAG. 5

Produsele lunii
Mobile Athlon XP
SuSE Linux 8.0
Dungeon Siege
ONI 2002

probleme.. PAG. 9

Olimpiada NAȚIONALĂ de Informatică
~ONI 2002, Brăila~

Concursul de informatică ȘTEFAN ODOBLEJA
~Craiova, județul Dolj~

Concursul de informatică URMAȘII LUI MOISIL
~Centrul de excelență, Iași~

40 de probleme propuse spre rezolvare la Olimpiada Națională și alte două concursuri interjudețene

concurs.. PAG. 33

Rezolvările problemelor celei de-a cincea runde

Clasamentul final

tehno.. PAG. 38

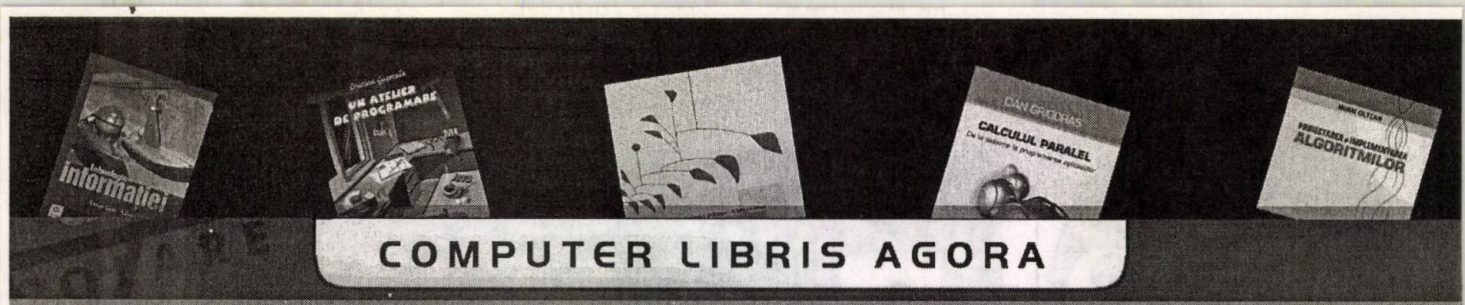
Tipuri de REȚELE
Mihai SCORȚARU
~Rețele de calculatoare~

serial.. PAG. 40

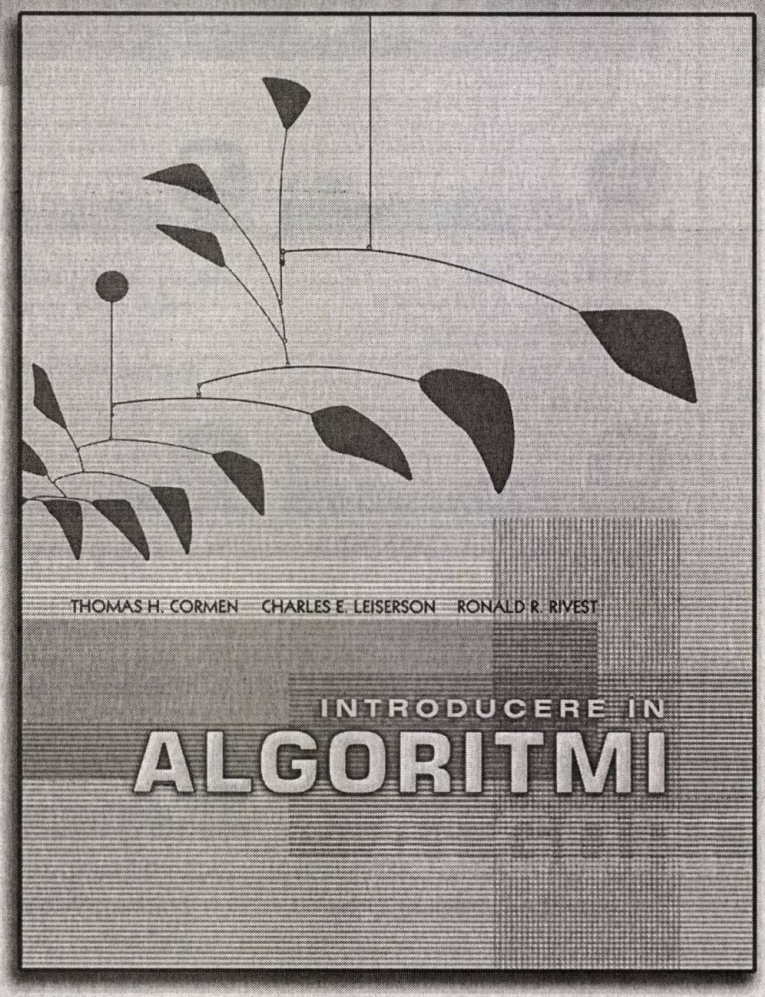
INSTRUMENTE de programare NESECVENȚIALĂ
Anca VASILESCU
Oana GEORGESCU

Tehnici de CRIPTARE
Claudiu SOROIU

Redacția GInfo vă urează o Vacanță plăcută!!!



COMPUTER LIBRIS AGORA



Cartea se adresează, în primul rând, celor care studiază la nivel elementar sau academic algoritmi și structuri de date. Deoarece tratează aspecte teoretice și practice, legate de proiectarea algoritmilor, este la fel de binevenită elevilor, studenților, profesorilor și autodidacților.
Introducere în algoritmi acoperă atât metodele clasice cât și metodele moderne ca analiza amortizată și algoritmii paraleli.
Prezentarea matematică, riguroasă, este atent detaliată în așa fel încât să fie accesibilă tuturor cititorilor.
Capitolele încep cu elemente de bază, avansând spre subiecte mai complexe.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

Atinși de GIGANT

În zilele de 18 și 19 aprilie a avut loc prima conferință RoEduNet. La eveniment au participat o mulțime de persoane importante din domeniul tehnologiei comunicațiilor, atât din România, cât și din afara ei. Cu ocazia acestei conferințe au fost anunțate câteva noutăți și proiecte care au sau vor avea un impact semnificativ asupra dezvoltării rețelei academice din țara noastră.

Despre conferință ar fi puține de spus: organizarea a fost bună, au apărut probleme inerente oricărui început, unele prezentări au fost interesante, altele extrem de plictisitoare, unele bine concepute, altele care lăsau de dorit.

Cel mai important este faptul că o astfel de conferință a avut loc, că au fost prezente personalități marcante cu putere de decizie la nivel european, că a fost un real succes etc.

Cel mai important anunț a fost faptul că *RoEduNet* este acum conectată la noua rețea europeană *Géant*, o rețea cu o capacitate de patru ori mai mare decât *Abilene*, rețeaua echivalentă de pe continentul nord-american.

Géant a apărut la sfârșitul anului 2001 și are ca scop crearea unei rețele pan-europene care să beneficieze de avantajele dezvoltării tehnologiei comunicațiilor. La ora actuală, prin intermediul acestei rețele, sunt interconectate 32 de rețele academice naționale, printre care și *RoEduNet*.

Nouă dintre circuitele internaționale din rețeaua *Géant* au o viteză de transfer de 10 Gbps. Pe alte unsprezece circuite viteza este de 2,5 Gbps. În acest moment *România* are o legătură de 155 Mbps.

Merită amintit faptul că *România* a fost prima țară care a cerut mărirea capacității legăturii externe. Inițial, legătura dintre *RoEduNet* și restul rețelei *Géant* avea o viteză de transfer de 33 *Mbps*.

Această viteză superioară (chiar dacă este enormă față de cei câțiva *Mbps* din anii trecuți) nu este atât de semnificativă precum faptul că *România* "a apărut pe harta" rețelei *Géant*. Prezența pe hartă este o dovadă clară a alinierii țării noastre la standardele europene, cel puțin în acest domeniu.

Dezvoltarea rețelei *Géant* este finanțată în mare parte de *Uniunea Europeană* care a alocat pentru acest proiect 80.000.000 *EURO*. La conferința de la *Cluj*, proiectul *Géant* a fost prezentat de către **Bernhard Fabianek**, reprezentant al *Comisiei Europene*.

Alături de prezentarea referitoare la rețeaua *Géant*, participanții la conferință au putut afla detalii referitoare la proiecte. Reprezentanții *Ministerului Educației și Cercetării* au prezentat diferite proiecte pentru cele două rețele academice din *România*: *RoEduNet* și **RNC** (*Rețeaua Națională pentru Cercetare*).

În *România* se va încerca impunerea unor tarife acceptabile pentru cercetare, educație și cultură. De asemenea, se va pune accentul pe *software*-ul educațional care va putea fi folosit în predarea diferitelor discipline cum ar fi matematică, fizică, chimie, biologie, limbi străine. Din nefericire, chiar informatica a fost omisă... Să sperăm că a fost doar o "scăpare"...

Alte proiecte importante prevăd unificarea celor două rețele academice, informatizarea învățământului din mediul rural (***prioritate guvernamentală!***), dotarea școlilor cu tehnică de calcul, manuale experimentale în format electronic etc.

O mare parte a timpului a fost alocată prezentărilor științifice. Din nou, unele au fost foarte interesante, altele nereușite. Numărul mare al lucrărilor prezentate este o dovadă a interesului acordat acestei conferințe de către reprezentanții mediului academic.

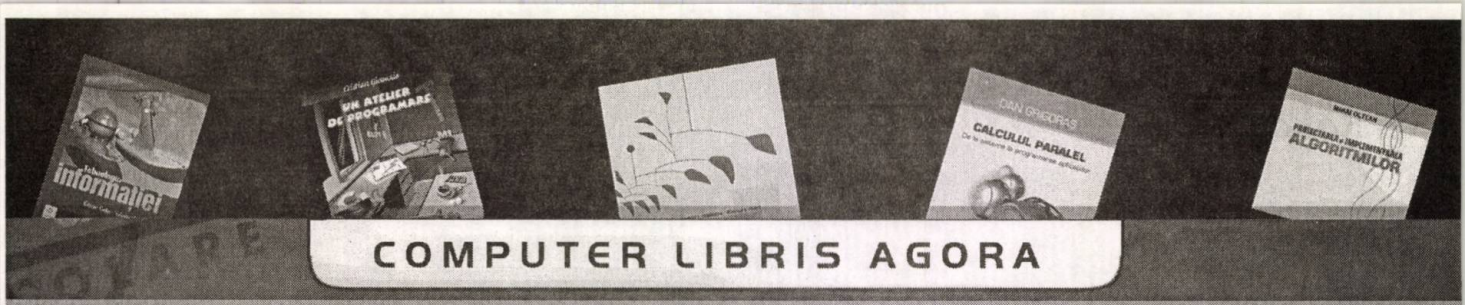
A avut loc și o expoziție în cadrul căreia firmele interesate au avut ocazia să își prezinte ultimele produse.

În sfârșit, *totul e bine când se termină cu bine*. Conferința a fost apreciată de oaspeții străini (chiar dacă au existat mici probleme tehnice), iar *România* a fost primită în familia europeană (cel puțin din punct de vedere academic).

Un rol foarte important în organizarea conferinței l-au avut *Universitatea Tehnică din Cluj* și filiala județeană *Cluj* a *RoEduNet*. Domnul **prof. dr. Pusztai Kálmán**, decan al *Facultății de Automatizări și Calculatoare* din cadrul universității mai sus amintite și director al filialei județene a *RoEduNet*, a fost cel care a coordonat pregătirile necesare în vederea bunei desfășurări a conferinței.

Felicităm organizatorii și le urăm succes în realizarea planurilor legate de dezvoltarea *RoEduNet*-ului.

Mihai Scorțaru



COMPUTER LIBRIS AGORA

Cartea este o culegere de studii de caz, construite pe temelia problemelor concursurilor Southeastern European Regional Programming Contest din perioada 1995-1999, concursuri organizate sub egida Association for Computing Machinery.

Cartea se adresează atât începătorilor, cât și celor cu experiență în domeniul programării, în încercarea de a contura prin exemple imaginea unei lumi diverse și pasionante, dar pretențioase, aceea a înzestrării mașinii de calcul cu capacitatea de a rezolva corect și eficient.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

Mobile Athlon XP

Compania AMD a lansat pe data de 17 aprilie 2002 primul său procesor fabricat folosind o tehnologie de 0,13 microni. Deocamdată, noua tehnologie este utilizată doar pentru procesoare destinate calculatoarelor portabile, dar versiunile pentru desktop-uri și server-e vor fi disponibile în curând. Datorită performanțelor deosebite și noutăților tehnologice introduse, AMD Mobile Athlon XP a devenit produsul lunii mai.

Introducere

Până acum, toate procesoarele *AMD* au fost fabricate folosind o tehnologie de 0,18 µm. Se știa că se dorea trecerea la tehnologia de 0,13 µm care permite construirea de procesoare cu dimensiuni reduse, frecvențe mai mari și costuri mai mici. Noile procesoare erau cunoscute doar sub numele de cod *Thoroughbred*.

AMD folosea denumiri diferite pentru procesoarele destinate diverselor categorii de sisteme. Astfel, procesoarele destinate *desktop*-urilor erau denumite *Athlon XP* (sufixul *XP* provine de la ***eXtreme Performance*** dar, de fapt, a fost ales datorită faptului că *Windows XP*, cel mai nou sistem de operare produs de *Microsoft*, conține același sufix, chiar dacă semnificația sa este diferită: ***eXPerience***). Procesoarele destinate calculatoarelor portabile erau denumite *Athlon* 4 (sufixul 4 a fost ales tot din motive comericiale pentru a elimina avantajul psihologic al procesoarelor *Pentium* 4). În sfârșit, procesoarele destinate *server*-elor erau denumite *Athlon MP* (sufixul *MP* are semnificația ***MultiProcessor*** și este folosit doar pentru a sugera o diferență (care în realitate nu există) între aceste procesoare mai scumpe și variantele mai ieftine pentru *desktop*-uri; practic, deși cele două tipuri de procesoare aveau o structură aproape identică, un *Athlon XP* costa cu 50$ mai puțin decât un *Athlon MP* care funcționa la aceeași frecvență.

Se pare că cel mai mare efect psihologic l-a avut denumirea *Athlon XP*; mulți utilizatori considerau faptul că, pentru un sistem nou, prezența sufixului *XP*, atât în denumirea procesorului, cât și în cea a sistemului de operare, indică o compatibilitate mai mare între componentele *hardware* și *software* ale sistemului.

Chiar dacă această explicație este total lipsită de logică, efectul psihologic a fost foarte mare și *AMD* a reușit să vândă un număr record de procesoare *Athlon XP*.

Ca urmare, compania a decis să folosească această strategie pentru toate categoriile de procesoare. Astfel, denumirea procesoarelor destinate calculatoarelor portabile a fost modificată din *Mobile Athlon* 4 în *Mobil Athlon XP*. Este foarte probabil ca o astfel de redenumire să se producă și în momentul lansării procesoarelor pentru *server*-e care vor fi fabricate pe baza tehnologiei de 0,13 µm.

Caracteristici

Pe lângă tehnologia de fabricație, există numeroase alte îmbunătățiri care au fost aduse de noul procesor.

În primul rând, *Mobile Athlon XP*, care folosește o frecvență a magistralei de 266 MHz, este primul procesor destinat calculatoarelor portabile .

La fel ca și în cazul procesoarelor *Athlon XP* destinate *desktop*-urilor, *AMD* nu folosește frecvența procesorului pentru a-i caracteriza performanțele, ci un indice de performanță care specifică frecvența pe care ar fi trebuit să o aibă un procesor din prima generație *Athlon* pentru a avea performanțe comparabile.

Astfel, noile procesoare apar în versiunile *Mobile Athlon XP* 1400+ și *Mobile Athlon XP* 1500+. Ele funcționează la o frecvență reală de 1,26 GHz, respectiv 1,33 GHz.

Până la sfârșitul lunii iunie vor fi disponibile două noi variante: *Mobile Athlon XP* 1400+ și *Mobile Athlon XP* 1500+, care vor funcționa, probabil, la frecvențe de 1,4 GHz, respectiv 1,47 GHz.

Celelalte caracteristici sunt identice cu cele ale procesoarelor *Athlon* 4: tehnologia *PowerNow*! pentru mărirea duratei de viață a bateriei, tehnologia *QuantiSpeed* care îmbunătățește performanțele sistemului dacă se folosește sistemul de operare *Windows XP*, seturile de instrucțiuni *3DNow*!, *SSE* etc.

Compania *AMD* vinde procesorul *Mobile Athlon XP* 1400+ la prețul de 190$, în timp ce versiunea *Mobile Athlon XP* 1500+ este comercializată la prețul de 250$. Un procesor *Mobile Athlon XP* 1600+ va costa 380$, iar un *Mobile Athlon XP* 1700+ va putea fi achiziționat la prețul de 489$.

Aceste prețuri sunt valabile pentru comenzi de minimum 1000 de bucăți; distribuitorii direcți ai companiei *AMD* pot beneficia de diverse reduceri.

SuSE Linux 8.0

Versiunea 8.0 a distribuției Linux oferite de compania SuSE aduce o serie de îmbunătățiri și noi facilități. Instalarea noului sistem de operare durează foarte puțin, iar interfața grafică a programului de instalare permite efectuarea acestei operații chiar și de către utilizatorii Linux neexperimentați.

Introducere

Noul sistem de operare *SuSE Linux* 8.0 aduce o serie de îmbunătățiri importante în ceea ce privește gama facilităților oferite, securitatea și ușurința de instalare și utilizare.

SuSE Linux 8.0 este una dintre cele mai complete distribuții *Linux*, concurând cu succes cu distribuțiile "rivale" *RedHat* și *Mandrake*. Recent, au fost lansate pe piață noi versiuni ale sistemelor de operare "concurente" și anume *RedHat Linux* 7.3 și *Mandrake Linux* 8.2.

Există două variante ale noii distribuții și anume *SuSE Linux* 8.0 *Personal* și *SuSE Linux* 8.0 *Professional*. Prima dintre ele costă 39.95$ și cuprinde trei *CD*-uri și două manuale de utilizare (*Basics* și *Applications*). Cea de-a doua costă 79.95$ și cuprinde șapte *CD*-uri, un *DVD* și trei manuale de utilizare (*Basics*, *Applications* și *The Reference*).

Instalare

În principiu, câteva *click*-uri sunt suficiente pentru ca *SuSE Linux* 8.0 să fie instalat pe calculator. Întregul proces de instalare poate dura doar 15 minute. În alte câteva minute sistemul de operare este configurat; astfel, instalarea sistemului de operare și a altor 350 de programe este realizată în aproximativ 30 de minute. Aceste performanțe sunt obținute datorită programului de instalare și configurare *YaST2*.

Pentru a putea utiliza *SUSE Linux* 8.0 PC-ul trebuie să fie echipat cu un procesor 486*DX* sau mai nou, 64 MB de memorie (dacă instalarea se realizează în mod text este suficientă o cantitate de memorie de 48 MB), cel puțin 400 MB de spațiu liber pe disc (se recomandă peste 2 GB).

Programul de instalare permite folosirea, pe același calculator, a sistemelor de operare *Linux* și *Windows*. Partițiile *Windows* sunt recunoscute de sistemul de operare și pot fi utilizate foarte ușor.

În timpul instalării se creează partițiile necesare sistemului de operare *Linux* fără ca datele *Windows* să fie alterate.

Prezentare

SuSE Linux 8.0 folosește cele mai noi versiuni pentru componentele cheie ale sistemului. Astfel, se folosește versiunea 2.4.18 a *kernel*-ului *Linux*, versiunea 3.0 a pachetului *KDE*, precum și pachetele *GNOME* 1.4.1 *RC* 1 (*Release Candidate* 1 - versiune *Beta*), *SuSE Firewall* 2 etc.

Noua distribuție este prima care include versiunea 3.0 a mediului *KDE* care ușurează foarte mult utilizarea acestui sistem de operare. *KDE* 3.0 introduce o serie de inovații care scurtează timpul necesar efectuării anumitor operații.

Distribuția conține și pachetul de aplicații pentru birou *StarOffice* 5.2. Acest pachet conține, pe lângă obișnuitul editor de texte, module care permit lucrul cu tabele, crearea de prezentări etc. De asemenea, *StarOffice* 5.2 pune la dispoziția utilizatorilor un client de *mail*, un editor *HTML*, un program pentru desenare, o aplicație pentru procesarea imaginilor și multe altele. Pe lângă *StarOffice* sunt puse la dipoziția utilizatorilor alte programe pentru birou cum ar fi *kOffice* sau cele din pachetul *GNOME*.

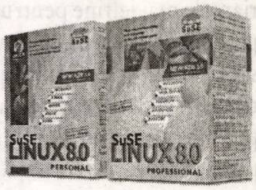
SuSE Linux 8.0 conține o suită impresionantă de aplicații *multimedia* care permit utilizarea și prelucrarea fișierelor video și audio.

Pentru cei care doresc să prelucreze fotografii este pus la dispoziție programul *gphoto* care permite transferul imaginilor din camerele digitale pe *PC*-uri. *Kooka*, programul pentru *scanner*-e, permite scanarea fotografiilor analogice și păstrarea lor pe *PC*-uri. În sfârșit, aplicația *GIMPS* vă permite să manipulați imaginile pentru a obține colecția de fotografii dorită.

Pentru mult așteptatele clipe de amuzament, *SuSE Linux* 8.0 vă pune la dispoziție o colecție impresionantă de jocuri. Câteva dintre acestea sunt *Racer* (joc de curse), *FlightGear* (simulator de avion), *TuxType* (deprinderea utilizării tastaturii) etc.

Concluzie

SuSE Linux 8.0 este una dintre cele mai complete și mai ușor de utilizat distribuții *Linux*. În privința ușurinței de instalare și utilizare, este un rival de temut pentru *Mandrake*, despre care se spune că este cel mai "prietenos" sistem de operare *Linux*.



Dungeon SIEGE

Pe data de 5 aprilie a fost lansat un nou joc care, potrivit companiei Microsoft, se dorește a fi cel mai interesant RPG disponibil în acest moment. Calitatea deosebită a graficii, precum și atmosfera jocului, vor atrage, cu siguranță, o mulțime de pasionați ai acestui gen.

Dungeon Siege a fost creat sub îndrumarea lui *Chris Taylor*, cel care s-a aflat la conducerea echipei care a realizat un alt joc celebru și anume *Total Anihilation*.

Primul lucru care îl impresionează pe jucător este calitatea extraordinară a graficii. Lumea virtuală a jocului este realizată prin intermediul unor peisaje *3D* extraordinare care oferă priveliști deosebite.

Peisajul poate fi admirat din diverse perspective, jucătorul putând alege să privească din orice poziție dorește; animațiile sunt fluide și au un realism deosebit.

Au fost eliminate neplăcerile cauzate de pauzele necesare încărcării anumitor niveluri în alte jocuri. În *Dungeon Siege* nu va trebui să așteptați decât foarte puțin, la începutul jocului.

O altă inovație este eliminarea aproape totală a caracteristicilor personajelor. La începutul jocului va trebui să alegeți doar numele, sexul și portretul personajului, fără a fi posibilă alegerea altor atribute. Personajele se vor dezvolta pe măsură ce își folosesc abilitățile.

De exemplu, dacă doriți să obțineți un bun vrăjitor, atunci va trebui să îi cereți personajului să folosească vrăji până în momentul în care o va face foarte bine.

Interfața jocului este foarte intuitivă, acesta putând fi controlat foarte ușor. Există o mulțime de taste care permit efectuarea rapidă a unor acțiuni relativ complicate; de asemenea, personajele pot fi controlate foarte ușor cu ajutorul *mouse*-ului.

Jucătorul este reprezentat de un singur personaj dar, pe parcursul jocului, alte personaje i se pot alătura (voluntar sau contra cost). Astfel, echipa de eroi cu ajutorul cărora vă veți aventura în acestă lume extraordinară poate fi formată din cel mult opt personaje; pot fi alese diverse formații de luptă sau diferite formații de marș.

Probabil, cel mai interesant și mai neobișnuit personaj care se poate alătura echipei este măgarul. Acesta are un rol esențial în joc, deoarece poate transporta o mulțime dintre obiectele la care celelalte personaje ar fi fost nevoite să renunțe datorită faptului că nu mai aveau posibilitatea să le transporte.

Din păcate, calitatea inteligenței artificiale cu care sunt dotați monștrii pe care îi veți întâlni lasă de dorit. Aceștia se aruncă direct în luptă și, în majoritatea cazurilor, cad în toate capcanele pe care jucătorii și le pot imagina.

Totuși, ei sunt destul de numeroși și au șanse mari de succes în cazul în care jucătorul alege să lupte simultan cu toți atacatorii. De obicei, monștrii trebuie atrași, unul câte unul, în diferite locuri în care pot fi ușor eliminați. Monștrii dintr-un grup nu colaborează între ei; fiecare se avântă în luptă pe cont propriu și astfel sunt mai ușor de învins.

Bineînțeles, nici unui *RPG* care se respectă nu îi poate lipsi opțiunea *multiplayer*. *Microsoft* nu a putut ignora acest element de bază și oferă jucătorilor posibilitatea de a se confrunta cu adversari umani sau de a parcurge jocul împreună cu aceștia.

În concluzie, calitatea deosebită a graficii, precum și cele câteva inovații prezente în joc, fac ca *Dungeon Siege* să merite atenția tuturor jucătorilor. Din nefericire, există unele elemente care îl împiedică să devină cel mai extraordinar *RPG* de până acum: acțiunea liniară și inteligența redusă a adversarilor.

Pentru a vă putea bucura de peisajele remarcabile prezente în acest joc va trebui să aveți la dispoziție un calculator cu procesor *Pentium* II cu frecvența de 333 MHz (sau mai bun), o dimensiune a memoriei *RAM* de 128 MB, o placă video cu cel puțin 16 MB, un *CD-ROM* care să poată citi la viteza de 1,2 MB/s (8×) și pe discul hard să aveți disponibil un spațiu liber de cel puțin 1 GB.

ONI 2002

Site-ul oficial al Olimpiadei Naționale de Informatică din acest an poate fi găsit la adresa www.eduitbr.ro. La această adresă veți putea găsi o mulțime de informații referitoare la ediția din acest an a concursului. Site-ul este foarte bine realizat, are un aspect plăcut dar, din nefericire, unele informații lipsesc.

Site-ul dedicat ediției 2002 a *Olimpiadei Naționale de Informatică* are un conținut diversificat, numeroasele secțiuni oferind informații din diferite domenii de interes.

Rubricile pe care le puteți vizita sunt: *Brăila on-line*, *Sponsori*, *Olimpiada Live*, *Program*, *Simpozion*, *Comisia Centrală*, *Participanți*, *Subiecte*, *Rezultate*, *ONI by Net*, *Carte de Oaspeți*, *Contact* și *Info Site*.

Cei care sunt interesați de diverse detalii privitoare la locul desfășurării olimpiadei din acest an vor găsi în secțiunea *Brăila on-line* o mulțime de informații referitoare la orașul gazdă. Sunt disponibile foarte multe imagini care prezintă diferite obiective turistice ale orașului.

În ultimii ani a devenit evident faptul că buna desfășurare a unei olimpiade depinde foarte mult de sponsori. Aceștia nu au lipsit nici la *ONI* 2002, iar pe site li s-a dedicat o rubrică specială. Aici veți putea găsi denumirile firmelor care au contribuit la organizarea acestui concurs.

În premieră, ediția 2002 a *Olimpiadei Naționale de Informatică* a fost transmisă în direct prin *Internet*. Pentru vizionarea unor imagini din timpul concursului a fost creată rubrica *Olimpiada Live* prin intermediul căreia puteau fi vizionate imagini cu cele mai interesante momente ale olimpiadei.

Rubrica *Program* descrie modul în care s-a desfășurat olimpiada. Sunt precizate reperele orare pentru diferite activități ale elevilor, ale profesorilor însoțitori sau ale membrilor *Comisiei Centrale*.

O altă premieră a fost desfășurarea unei serii de prezentări susținute de unii membri ai comisiei. Titlurile lucrărilor prezentate apar în cadrul rubricii *Simpozion*.

Lista membrilor *Comisiei Centrale* poate fi găsită în rubrica dedicată acesteia. O rubrică asemănătoare este dedicată concurenților; aici sunt prezente numele tuturor elevilor care au participat la *ONI* 2002.

Probabil cea mai interesantă rubrică a *site*-ului este cea în care sunt puse la dispoziție enunțurile problemelor de concurs. Aici veți găsi problemele propuse spre rezolvare în primele două zile ale concursului. Din nefericire, nu am reușit să găsim pe acest *site* enunțurile problemelor de la barajul de selecție a membrilor lotului național de informatică.

O altă rubrică foarte interesantă este cea în care sunt prezentate clasamentele concursului. Sunt disponibile clasamentele de la fiecare clasă dar, din nou, barajul de selecție este omis, clasamentul obținut în urma desfășurării acestuia lipsind de pe *site*.

O mulțime de imagini interesante de la locul desfășurării olimpiadei pot fi privite în cadrul rubricii *Galerie Foto*.

Din nou, concursul "real" de la *Brăila* a fost completat de varianta *on-line* a olimpiadei. *ONI by Net* s-a desfășurat în paralel cu olimpiada propriu-zisă; la acest concurs au putut participa toți cei care nu au reușit să se califice la *ONI* 2002. Pe *site*-ul olimpiadei se află un *link* spre *site*-ul dedicat concursului *online*; acesta este disponibil la adresa **http://liceu.oni.pcnet.ro**.

Anumite rubrici obligatorii pentru un *site* care se respectă nu puteau să lipsească. Astfel, vi se pune la dispoziție rubrica în care vă puteți exprima diferite păreri referitoare la olimpiadă sau la *site* (*Cartea de Oaspeți*); de asemenea sunt prezente și informațiile de contact, precum și diverse alte informații referitoare la acest *site*.

În concluzie, chiar dacă există unele lipsuri, *site*-ul dedicat ediției 2002 a *Olimpiadei Naționale de Informatică* se ridică la nivelul așteptărilor. Evident, există anumite aspecte care ar putea fi îmbunătățite. De exemplu, nu există pagini dedicate enunțurilor problemelor. Acestea pot fi studiate doar dacă se descarcă anumite arhive care conțin enunțurile în format `.HTML` și `.DOC`. De asemenea, clasamentele sunt disponibile doar în format `.XLS`.

În rest, ar fi puține de reproșat acestui *site*. Grafica este simplă, dar atrăgătoare, *site*-ul nefiind încărcat cu ornamente care nu își au rostul. Realizatorii săi merită felicitați pentru efortul depus în crearea acestuia.

ONI 2002, Brăila

Olimpiada NAȚIONALĂ de Informatică

În perioada 27 aprilie-3 mai a avut loc la Brăila ediția din acest an a Olimpiadei Naționale de Informatică. Vă prezentăm în continuare enunțurile celor 24 de probleme propuse spre rezolvare la ONI 2002.

Clasa a IX-a

P050201: Pentagon

(lect. univ. Ovidiu Domșa, Alba Iulia)

În urma bombardamentelor din 11 septembrie 2001, unul dintre pereții clădirii *Pentagonului* a suferit daune majore. Imaginea codificată a peretelui avariat se reprezintă sub forma unei matrice cu m linii și n coloane ale cărei elemente sunt numere întregi din mulțimea {0, 1}. Valoarea 1 are semnificația "zid intact", în timp ce semnificația valorii 0 este "zid avariat".

Sumele alocate de *Bin Laden* pentru refacerea *Pentagonului* vor fi donate celor care îi vor ajuta pe americani să refacă această clădire prin plasarea, pe verticală, a unor blocuri de diverse înălțimi și lățime 1, în zonele care au fost avariate.

Pentru o structură dată a unui perete din clădirea *Pentagonului*, determinați numărul minim de blocuri necesare pentru refacerea clădirii.

Date de intrare

Fișierul de intrare `PENTAGON.IN` conține, pe prima linie, dimensiunile m și n ale peretelui clădirii, iar pe următoarele m linii câte n valori din mulțimea {0, 1}, neseparate prin spații.

Date de ieșire

Fișierul de ieșire `PENTAGON.OUT` va conține, pe câte o linie, secvențe de forma k nr, unde k indică înălțimea unui bloc, iar nr este numărul de blocuri de înălțime k. Cele două numere vor fi separate prin câte un spațiu, iar liniile vor fi ordonate crescător, în funcție de valoarea k.

Restricții și precizări

- $1 \le m, n \le 200$;
- în fișierul de ieșire nu vor apărea linii de forma k 0.

Exemplu

`PENTAGON.IN`

```
5 10
1110000111
1100001111
1000000011
1111101111
1110000111
```

`PENTAGON.OUT`

```
1 7
2 1
3 2
5 1
```

Timp de execuție: 1 secundă/test

P050202: Pod

(prof. Marinel Șerban, Iași)

Între două maluri ale unei văi adânci s-a construit un pod suspendat format din N bucăți de scândură, legate cu liane. Vom considera că scândurile sunt numerotate de la 1 la N, începând de pe malul pe care ne aflăm. În timp, unele bucăți de scândură s-au deteriorat, iar altele chiar au dispărut. Pentru traversarea podului se știe că:

- se pot face pași doar de lungime 1, 2 sau 3;
- scândurile deteriorate sunt nesigure, deci pe ele și de pe ele se pot face doar pași de lungime 1;
- evident, nu se poate păși pe o scândură care lipsește.

Scrieți un program care să determine numărul de modalități de traversare a podului (mai exact, de a ajunge pe celălalt mal), precum și o soluție de traversare, dacă o astfel de soluție există.

Date de intrare

Prima linie a fișierului de intrare `POD.IN` conține numărul n al scândurilor. Primul număr de pe a doua linie este k și indică numărul scândurilor care lipsesc. Această linie mai conține numerele de ordine ale celor k scânduri. Primul număr de pe a treia linie este h și indică numărul scândurilor deteriorate. Linia mai conține numerele de ordine ale celor h scânduri. Numerele de pe o linie sunt separate prin spații.

Date de ieșire

Prima linie a fișierului de ieșire **POD.OUT** va conține numărul de posibilități de a traversa podul. În cazul în care podul nu poate fi traversat, linia va conține valoarea -1. Dacă podul poate fi traversat, pe a doua linie va fi descrisă o modalitate de traversare. Pe această linie se vor afla numerele de ordine ale scândurilor pe care se pășește, în ordinea în care acestea sunt parcurse.

Restricții și precizări

- $3 \le n \le 300$;
- $0 \le k, h \le n$;
- prima scândură nu poate lipsi;
- numerele de ordine ale scândurilor deteriorate și numerele de ordine ale scândurilor care lipsesc formează două mulțimi disjuncte;
- numărul posibilităților de traversare conține cel mult 80 de cifre.

Exemple

POD.IN	POD.OUT
5 0 0	24 3

POD.IN	POD.OUT
10 2 2 7 1 5	48 3 6 8

POD.IN	POD.OUT
6 2 2 4 1 3	-1

Timp de execuție: 1 secundă/test

P050203: Suma

(Florin Ghețu, București)

Considerăm șirul format din primele N numere naturale: $a_i = i$ pentru $i = 1, ..., N$. Fiecărui element al acestui șir i se asociază un semn (+ sau -).

Pentru o valoare S dată se cere să se determine cel mai mic număr N astfel încât, după asocierea semnelor, elementele obținute să aibă suma S.

Date de intrare

Fișierul **SUMA.IN** va conține un singur număr întreg care reprezintă suma care trebuie obținută după asocierea semnelor.

Date de ieșire

Fișierul de ieșire **SUMA.OUT** va conține, pe prima linie, numărul N al elementelor șirului. Pe următoarele linii se vor afla indicii elementelor care au semn negativ, câte unul pe o linie.

Restricție

- $0 < S \le 100000$.

Exemplu

SUMA.IN	SUMA.OUT
12	7 1 7

Timp de execuție: 1 secundă/test

P050204: Becuri

(prof. Cristina Bărbieru, Timișoara)

Un panou publicitar, de formă dreptunghiulară conține becuri, unul lângă altul, aliniate pe linii și coloane. Fiecare linie și fiecare coloană are un comutator care schimbă starea tuturor becurilor de pe acea linie sau coloană, din starea în care se află în starea opusă. Inițial panoul are toate becurile stinse.

Să se realizeze un program care, acționând asupra unui număr minim de linii și coloane, aduce panoul din starea inițială, la o configurație dată, dacă acest lucru este posibil.

Date de intrare

Pe prima linie a fișierului de intrare **BECURI.IN** se află două numere naturale, m și n, separate printr-un spațiu, reprezentând numărul de linii, respectiv coloane, ale panoului. Pe următoarele m linii se află câte n numere, separate prin câte un spațiu, reprezentând configurația finală a panoului. Un bec aprins este codificat prin 1, în timp ce un bec stins este codificat prin 0.

Date de ieșire

Pe prima linie a fișierului de ieșire **BECURI.OUT** se vor afla indicii coloanelor asupra cărora s-a acționat. Cea de-a doua linie va conține indicii liniilor asupra cărora s-a acționat. Numerele de pe o linie vor fi separate prin câte un spațiu. Dacă nu se acționează asupra nici unei linii sau asupra nici unei coloane, pe linia corespunzătoare va fi scrisă doar valoarea 0. Numerotarea liniilor și a coloanelor începe de la 1. În cazul în care nu există nici o soluție, fișierul de ieșire va conține o singură linie pe care se va afla valoarea -1.

Restricție

- $1 \le m, n \le 100$.

Exemplu

BECURI.IN	BECURI.OUT
5 6 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 0 1 0 1 0 0 1 0 0 1 0 0 1 0	2 5 1 2 3

Timp de execuție: 1 secundă/test

P050205: Discuri

(Florin Ghețu, București)

Se dau N numere reale care reprezintă razele a N discuri. Considerăm că așezăm un disc în sistemul xOy dacă îl plasăm la o coordonată x pozitivă suficient de mare, tangent cu axa Ox și deasupra ei, apoi îl împingem spre Oy până când devine tangent cu Oy sau cu primul disc întâlnit, așezat anterior. În figura rezultată după așezarea tuturor discurilor în ordinea dată, unele dintre ele pot fi considerate ***dispensabile*** deoarece, prin eliminarea lor, nu se modifică lățimea totală a figurii rezultate, adică nici un disc nu se mai poate deplasa spre stânga. Sarcina voastră este de a scrie un program care identifică toate discurile dispensabile dintr-o configurație dată.

Date de intrare

Fișierul de intrare **DISCURI.IN** conține pe prima linie numărul N al discurilor, iar pe următoarele N linii câte un număr real care indică raza unui disc. Razele apar în ordinea amplasării discurilor.

Date de ieșire

Prima linie a fișierului de ieșire **DISCURI.OUT** va conține numărul K al discurilor dispensabile. Pe următoarele K linii se vor afla numerele de ordine ale discurilor dispensabile.

Restricție

- $1 \le N \le 1000$.

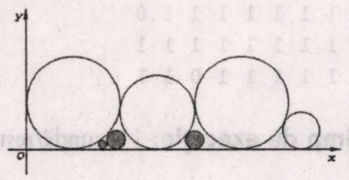
Exemplu

DISCURI.IN

```
7
4
0.1
0.5
3
0.5
4
1
```

DISCURI.OUT

```
3
2
3
5
```



cercurile hașurate sunt dispensabile

Timp de execuție: 1 secundă/test

P050206: Cod

(lect. univ. Ovidiu Domșa, Alba Iulia)

Transmiterea și memorarea informațiilor necesită diverse sisteme de codificare în vederea utilizării optime a spațiilor disponibile. Un sistem foarte des întâlnit este acela prin care unei secvențe de caractere i se asociază un număr.

Se consideră cuvintele formate numai cu literele mici ale alfabetului englez. Dintre toate aceste cuvinte le considerăm doar pe cele ale căror caractere sunt în ordine strict lexicografică (caracterul de pe orice poziție este strict mai mic decât un eventual succesor). Sistemul de codificare se obține astfel:

- se ordonează cuvintele în ordinea crescătoare a lungimilor lor;
- cuvintele de aceeași lungime se ordonează lexicografic;
- cuvinte sunt codificate prin numerotarea lor (în șirul obținut după ordonare):

```
1.      - a
2.      - b
...
26.     - z
27.     - ab
...
51.     - az
52.     - bc
...
83681.  - vwxyz
...
```

Dându-se un cuvânt, să se precizeze dacă poate fi codificat conform sistemului descris. În caz afirmativ să se precizeze codul său.

Date de intrare

Fișierul de intrare **COD.IN** conține o singură linie pe care se află un cuvânt.

Date de ieșire

În cazul în care cuvântul poate fi codificat, fișierul de ieșire **COD.OUT** va conține o singură linie pe care se va afla codul acestuia. În caz contrar, fișierul va conține doar valoarea 0.

Restricție

- un cuvânt conține cel mult 10 litere.

Exemple

COD.IN

```
bf
```

COD.OUT

```
55
```

COD.IN

```
aab
```

COD.OUT

```
0
```

COD.IN

```
vwxyz
```

COD.OUT

```
83681
```

Timp de execuție: 2 secunde/test

Clasa a X-a

P050207: Hotel

(prof. Doru Popescu Anastasiu, Slatina)

Departamentul administrativ al unui hotel are n angajați. Patronul hotelului hotărăște să schimbe costumele personalului din acest departament, astfel încât angajații care lucrează la etaje diferite să fie îmbrăcați în haine colorate diferit, iar cei care lucrează la același etaj să fie îmbrăcați în haine colorate la fel. Angajații sunt identificați printr-un cod unic, care constă într-un număr natural format din cel mult patru cifre.

Să se determine numărul modalităților de alegere a costumelor, astfel încât să fie respectate condițiile descrise

anterior. De asemenea, trebuie determinată și una dintre aceste modalități.

Date de intrare

Prima linie a fișierului de intrare **HOTEL.IN** conține numărul n al angajaților și numărul k al culorilor disponibile. Cele două numere sunt separate printr-un spațiu. Pe următoarele n linii se afla câte două numere naturale, separate printr-un spațiu, primul reprezentând codul, iar al doilea etajul asociat unui angajat.

Date de ieșire

Prima linie a fișierului de ieșire **HOTEL.OUT** va conține numărul modalităților de alegere a costumelor. Știind că fiecare culoare este codificată printr-un număr natural nenul mai mic sau egal cu k și că numerele asociate culorilor sunt distincte, în fișier se va scrie pe câte o linie (începând cu a doua) codul unei persoane și culoarea costumului, valori separate prin câte un spatiu. Ordinea aparitiei în fișierul de iesire va fi aceeași cu cea din fișierul de intrare. Dacă nu există nici o soluție, în fișier se va scrie o singură linie care va conține valoarea 0.

Restricții și precizări

- $1 \le n \le 1000$;
- $1 \le k \le 200$;
- hotelul are cel mult 200 de etaje.

Exemple

HOTEL.IN
```
4 5
123 2
35 1
430 2
13 0
```

HOTEL.OUT
```
60
123 1
35 2
430 1
13 3
```

HOTEL.IN
```
5 2
12 1
13 0
14 1
10 2
11 0
```

HOTEL.OUT
```
0
```

Timp de execuție: 1 secundă/test

P050208: Lac

(prof. Dan Grigoriu, București)

O zonă mlăștinoasă are forma dreptunghiulară, având nl linii și nc coloane. Ea este formată din celule cu latura de o unitate. O parte din acestea reprezintă *uscat*, iar altele reprezintă *apă*, uscatul fiind codificat cu 0, iar apa cu 1. Se dorește să se obțină un drum de pe malul de nord spre cel de sud, trecând doar pe uscat. Celulele cu apă pot fi transformate în uscat, parașutând într-un loc cu apă câte un ponton (o plută) de dimensiunea unei celule. Deoarece parașutarea este periculoasă, se dorește minimizarea numărului de parașutări. Pentru deplasare, dintr-o celulă se poate trece într-o celulă vecină (pe linie, coloană sau diagonală).

Scrieți un program care determină numărul minim de pontoane și coordonatele acestora.

Date de intrare

Fișierul de intrare **LAC.IN** conține, pe prima linie, numerele naturale nl și nc, separate printr-un singur spațiu. Pe următoarele nl linii se află câte nc valori binare, separate prin câte un spațiu, reprezentând configurația zonei (0 pentru uscat și 1 pentru apă).

Date de ieșire

Prima linie a fișierului de ieșire **POD.OUT** va conține numărul minim k al pontoanelor necesare. Pe următoarele k linii se vor afla câte două numere naturale, separate prin câte un spațiu, reprezentând linia, respectiv coloana, în care a fost amplasat unul dintre pontoane.

Restricție

- $1 \le nl, nc \le 100$.

Exemplu

LAC.IN
```
8 9
0 1 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1 1
1 0 1 1 1 0 1 1 1
1 1 0 0 1 1 0 1 1
1 1 1 1 1 1 1 0 1
1 1 1 1 1 1 1 1 0
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 0 1 1
```

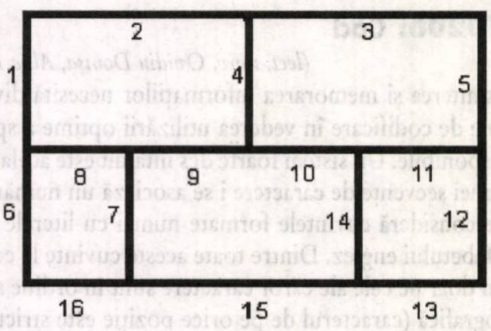
LAC.OUT
```
2
4 5
7 8
```

Timp de execuție: 1 secundă/test

P050209: Logic

(Marius Andrei, București)

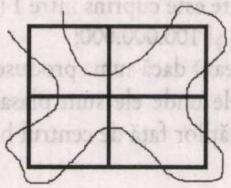
Într-o zi a venit la mine un coleg din clasa a X-a și mi-a propus un joc de logică. Mi-a arătat următoarea figură:



Pe această figură am numerotat segmentele, pentru a fi mai clară noțiunea de segment. Eu am la dispoziție un creion care se află pe hârtie în zona exterioară și trebuie să trasez curbe pe foaie, fără să ridic creionul, astfel încât să

trec prin toate segmentele (fără a atinge extremitățile) exact o dată. La sfârșit trebuie să mă aflu tot în exterior. Liniile (curbele) mele se pot intersecta. Am început și am încercat de mai multe ori, dar n-am reușit.

Acum, când am mai crescut, am reușit să demonstrez că nu se poate, dar am observat că pentru alte figuri este posibil. Un exemplu în care problema are soluție este:



Eu mi-am dat seama de ce uneori se poate și alteori nu, dar vreau să văd dacă reușiți și voi. De aceea vă voi da câteva figuri și, pentru fiecare, trebuie să răspundeti cu DA sau NU la întrebarea: *se poate sau nu trasa o curbă având tipul descris mai sus?*

Date de intrare

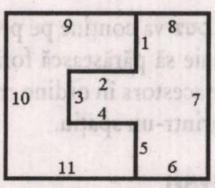
Fișierul de intrare **LOGIC.IN** are următoarea structură:

- pe prima linie se află numărul T al figurilor;
- pe următoarele T grupuri de linii se află datele corespunzătoare celor T figuri, după cum urmează:
  - pe prima linie a grupului se află valoarea N care reprezintă numărul de linii și de coloane ale matricei pătratice care descrie figura;
  - pe următoarele N linii se află câte N numere, separate printr-un spațiu (numere întregi cuprinse între 0 și 255), care reprezintă elementele matricei.

Această matrice codifică figura astfel: definim o zonă ca fiind o parte continuă a matricei care conține același număr și este aria maximă cu această proprietate; altfel spus, două căsuțe alăturate (care diferă la o singură coordonată printr-o unitate) care conțin același număr, se lipesc. Cu alte cuvinte, în interiorul zonei nu vor exista segmente. La aceste zone se mai adaugă și exteriorul matricei ca fiind o nouă zona. În matrice pot fi zone cărora le corespunde același număr, dar care "nu se ating" și se consideră a fi zone diferite. Segmentele sunt segmente de dreaptă care separă două zone și au lungimea maximă. De exemplu, figura definită prin

```
3
1 1 2
1 2 2
1 1 2
```

este următoarea:



Se observă că între zonele 1 și 2 există cinci segmente, între zona 1 și exterior există trei segmente, iar între zona 2 și exterior există tot trei segmente. Trebuie remarcat faptul că segmentul numerotat în figura anterioară cu 10 este compus din trei segmente mici, dar este considerat a fi un singur segment, conform definiției.

Date de ieșire

În fișierul **LOGIC.OUT** se vor scrie T linii, corespunzătoare celor T figuri. Pe fiecare linie va fi scris unul dintre mesajele DA sau NU, în funcție de răspunsul la întrebare.

Restricții

- $1 \le T \le 10$;
- $1 \le N \le 100$.

Exemplu

LOGIC.IN	**LOGIC.OUT**
2	DA
2	NU
1 2	
3 4	
4	
1 1 2 2	
1 1 2 2	
3 4 4 1	
3 4 4 1	

Timp de execuție: 1 secundă/test

P050210: Foto

(prof. Roxana Tâmplaru, Craiova)

Gigel, specialist în editare grafică pe calculator, se confruntă cu o problemă. El trebuie să aranjeze patru fotografii, disponibile în format electronic, într-o pagină de prezentare, astfel încât suprafața paginii să fie complet "acoperită" de cele patru fotografii și fără ca acestea să se suprapună în vreun fel. *Gigel* poate modifica dimensiunile inițiale ale fotografiilor, însă fără a deforma imaginile. Pentru aceasta el trebuie să păstreze neschimbat raportul dintre lungimea și înălțimea fotografiilor. Nu contează ordinea așezării fotografiilor, ele putând fi translatate oriunde în cadrul paginii, însă operațiile de rotație nu sunt permise.

Determinați pentru fiecare fotografie dimensiunile finale, cunoscându-se dimensiunile paginii, precum și dimensiunile inițiale ale fotografiilor.

Date de intrare

Prima linie a fișierului de intrare **FOTO.IN** conține două numere naturale l și h, separate printr-un spațiu, reprezentând lungimea, respectiv înălțimea paginii. Pe următoarele patru linii se află câte două numere naturale, separate printr-un spațiu, reprezentând lungimea, respectiv înălțimea, uneia dintre cele patru fotografii care trebuie aranjate.

Date de ieșire

Fișierul de ieșire **FOTO.OUT** va conține patru linii pe care se vor afla câte două numere naturale care reprezintă dimen-

siunile (lățime și înălțime) finale ale celor patru fotografii. Prima linie va corespunde primei fotografii din fișierul de intrare, a doua linie celei de-a doua fotografii etc.

Restricții și precizări

- dimensiunile paginii și ale fotografiilor sunt numere naturale cuprinse între 1 și 2000;
- dacă există mai multe soluții va fi determinată doar una dintre ele;
- există întotdeauna cel puțin o posibilitate de amplasare a fotografiilor.

Exemplu

FOTO.IN	FOTO.OUT
140 140	20 10
24 12	40 130
4 13	100 140
10 14	20 10
4 2	

Timp de execuție: 1 secundă/test

P050211: Balanță

(Mihai Stroe, București)

Gigel are o "balanță" mai ciudată pe care vrea să o echilibreze. De fapt, aparatul este diferit de orice balanță pe care ați văzut-o până acum. Balanța lui *Gigel* dispune de două brațe de greutate neglijabilă, fiecare având lungimea 15. Din loc în loc, pe aceste brațe se află cârlige, de care *Gigel* poate atârna greutăți distincte din colecția sa de G greutăți (numere naturale cuprinse între 1 și 25). *Gigel* poate atârna oricâte greutăți de orice cârlig, dar trebuie să folosească toate greutățile de care dispune.

Folosindu-se de experiența participării la *Olimpiada Națională de Informatică*, *Gigel* a reușit să echilibreze balanța relativ repede, dar acum dorește să știe în câte moduri poate fi ea echilibrată.

Cunoscând amplasamentul cârligelor și greutățile pe care *Gigel* le are la dispoziție, scrieți un program care calculează în câte moduri se poate echilibra balanța.

Date de intrare

Fișierul de intrare **BALANTA.IN** are următoarea structură:

- pe prima linie se află numărul C al cârligelor și numărul G al greutăților; cele două numere sunt separate printr-un spațiu;
- pe a doua linie se află C numere întregi distincte, separate prin spațiu, cu valori cuprinse între -15 și 15 inclusiv, reprezentând amplasamentele cârligelor față de centrul balanței; valoarea absolută a numerelor reprezintă distanța față de centrul balanței, iar semnul indică brațul balanței pe care se află cârligul, `'-'` pentru brațul stâng și `'+'` pentru brațul drept;
- pe a treia linie se află G numere naturale distincte, cuprinse între 1 și 25, reprezentând valorile greutăților pe care *Gigel* le va folosi pentru a echilibra balanța.

Date de ieșire

Fișierul de ieșire **BALANTA.OUT** conține o singură linie, pe care se află un număr natural M, numărul de variante de plasare a greutăților care duc la echilibrarea balanței.

Restricții și precizări

- $2 \le C, G \le 20$;
- numărul de variante este cuprins între 1 (există întotdeauna cel puțin una) și 100.000.000;
- balanța se echilibrează dacă suma produselor dintre greutăți și coordonatele unde ele sunt plasate este 0 (suma momentelor greutăților față de centrul balanței este 0).

Exemplu

BALANTA.IN	BALANTA.OUT
2 4	2
-2 3	
3 4 5 8	

Timp de execuție: 1 secundă/test

P050212: Aliniere

(prof. Moț Nistor, Brăila)

În armată, o companie este alcătuită din n soldați. La inspecția de dimineață soldații stau aliniați în linie dreaptă în fața căpitanului. Acesta nu e mulțumit de ceea ce vede; e drept că soldații sunt așezați în ordinea numerelor de cod din registru, dar nu în ordinea înălțimii. Căpitanul cere câtorva soldați să iasă din rând, astfel ca cei rămași, fără a-și schimba locurile, doar apropiindu-se unul de altul (pentru a nu rămâne spații mari între ei) să formeze un șir în care fiecare soldat vede privind de-a lungul șirului, cel puțin una din extremități (stânga sau dreapta). Un soldat vede o extremitate dacă între el și capătul respectiv nu există un alt soldat cu înălțimea mai mare sau egală cu a lui.

Scrieți un program care determină, cunoscând înălțimea fiecărui soldat, numărul minim de soldați care trebuie să părăsească formația, astfel ca șirul rămas să respecte condiția impusă de căpitan.

Date de intrare

Pe prima linie a fișierului de intrare **ALINIERE.IN** se află numărul n al soldaților din șir, iar pe linia următoare se află un șir de n numere reale, cu maximum cinci zecimale fiecare, separate prin spații. Al k-lea număr de pe această linie reprezintă înălțimea soldatului cu codul k.

Date de ieșire

Fișierul **ALINIERE.OUT** va conține pe prima linie numărul soldaților care trebuie să părăsească formația, iar pe linia următoare, codurile acestora în ordine crescătoare, separate două câte două printr-un spațiu.

Restricții și precizări

- înălțimile soldaților sunt numere reale cuprinse între 0,5 și 2,5;

- $2 \le n \le 1000$;
- dacă există mai multe soluții posibile, se va alege doar una dintre ele.

Exemplu

ALINIERE.IN

```
8
1.86 1.86 1.30621 2 1.4 1 1.97 2.2
```

ALINIERE.OUT

```
4
1 3 7 8
```

Timp de execuție: 1 secundă/test

Clasele a XI-a și a XII-a

P050213: Arbore

(prof. Emanuela Cerchez, Iași)

Se consideră un arbore cu N vârfuri, numerotate de la 1 la N. Scrieți un program care adaugă un număr minim de muchii, astfel încât fiecare vârf al grafului obținut să aparțină exact unui singur ciclu.

Date de intrare

Prima linie a fișierului de intrare **ARBORE.IN** conține numărul N al vârfurilor arborelui. Pe următoarele N - 1 linii se află câte două numere, separate printr-un spațiu, care reprezintă extremitățile uneia dintre muchiile arborelui.

Date de ieșire

Prima linie a fișierului de ieșire **ARBORE.OUT** conține numărul Nr al muchiilor adăugate. Pe următoarele Nr - 1 linii se află câte două numere, separate printr-un spațiu, care reprezintă extremitățile uneia dintre muchiile adăugate. În cazul în care problema nu are soluție, fișierul de ieșire va conține o singură linie pe care se va afla valoarea -1.

Restricție

- $3 \le N \le 1000$.

Exemple

ARBORE.IN

```
4
1 2
2 3
2 4
```

ARBORE.OUT

```
-1
```

ARBORE.IN

```
7
1 2
1 3
3 5
3 4
5 6
5 7
```

ARBORE.OUT

```
2
6 7
2 4
```

Timp de execuție: 1 secundă/test

P050214: Decodificare

(Mugurel-Ionuț Andreica, București)

Serviciul Român de Informații a dat de urma unei organizații teroriste care își are sediul pe teritoriul țării noastre. Folosind cei mai pricepuți și mai bine antrenați spioni și ofițeri, *SRI* a reușit să identifice computerul principal al organizației teroriste. Dacă va reuși să acceseze informațiile din acest computer, *SRI* îi va putea aresta pe toți membrii organizației și va asigura menținerea păcii mondiale. Singura problemă este spargerea codului de acces. Tot ceea ce se știe despre acest cod este că el este reprezentat de către o permutare de lungime N. Specialiștii *SRI* au încercat diverse metode de a descoperi codul, însă tot ceea ce au reușit să obțină este un program care, transmițându-i-se ca parametru o permutare de lungime N, specifică în câte poziții codul de acces coincide cu aceasta.

Scrieți un program care, folosind programul-ajutător (reprezentat sub forma unui modul), determină codul de acces în computerul teroriștilor.

Date de intrare

Programul dumneavoastră nu va citi date din nici un fișier de intrare. El va apela întâi funcția `GetN` a modulului `PROG`, care va returna valoarea N - numărul de elemente ale permutării care trebuie descoperită.

Apoi va apela numai funcția `Check`, căreia îi va fi transmisă ca parametru, de fiecare dată, o permutare de lungime N. Această funcție va returna numărul de poziții în care permutarea coincide cu permutarea care trebuie descoperită. Programul dumneavoastră trebuie ca, după un număr finit de apelări ale funcției `Check`, să descopere permutarea căutată.

Date de ieșire

Programul dumneavoastră va trebui să tipărească în fișierul **DECOD.OUT** o permutare de lungime N. Toate cele N elemente vor fi tipărite pe prima linie a fișierului, fiind separate de câte un spațiu.

Restricție

- $5 \le N \le 256$.

Instrucțiuni pentru programatorii în C/C++

Programatorii în *C/C++* au la dispoziție *header*-ul **PROG.H**. În acest fișier sunt declarate următoarele funcții:

```
int GetN(void)
int Check(int p[256])
```

Funcția `Check` are ca parametru un vector cu elemente întregi. Ea va returna una dintre următoarele valori:

- valoarea -1 dacă primele N elemente (începând cu poziția 0) ale vectorului transmis ca parametru nu constituie o permutare de lungime N;
- în cazul în care permutarea este corectă, se returnează un număr cuprins între 0 și N care reprezintă numărul de

poziții în care permutarea de lungime N transmisă ca parametru coincide cu permutarea care trebuie descoperită.

Instrucțiuni pentru programatorii în Pascal

Modulul extern este implementat sub forma *unit*-ului **PROG**. În acest *unit* sunt declarate următoarele tipuri și funcții care vor fi folosite în programul vostru:

```
type perm=array[1..256] of Integer;
function GetN:Integer;
function Check(var permut:perm):Integer;
```

Funcția `Check` are ca parametru un vector de tipul `perm`. Ea va returna una dintre următoarele valori:

- valoarea -1 dacă primele N elemente (începând cu poziția 0) ale vectorului transmis ca parametru nu constituie o permutare de lungime N;
- în cazul în care permutarea este corectă, se returnează un număr cuprins între 0 și N care reprezintă numărul de poziții în care permutarea de lungime N transmisă ca parametru coincide cu permutarea care trebuie descoperită.

Exemplu

Să presupunem că permutarea căutată este `2 1 3 4 5`. La început va fi apelată funcția `GetN`; aceasta va returna valoarea `5`. Vor urma apeluri succesive ale funcției `Check`, până în momentul în care aceasta returnează valoarea `5`. O posibilă succesiune este:

- apel pentru permutarea `1 2 3 4 5` - se returnează `3`;
- apel pentru permutarea `3 2 1 4 5` - se returnează `2`;
- apel pentru permutarea `2 1 3 4 5` - se returnează `5`.

Permutarea corectă este scrisă în fișierul **DECOD.OUT**.

Timp de execuție: 1 secundă/test

P050215: SETI

(Mihai Pătrașcu, Craiova)

Se pare că, în sfârșit, căutătorii vieții extraterestre au descoperit ceva! În cursul proiectului *SETI@home* a fost izolată o secvență care ar putea reprezenta un semnal de la alte forme de viață inteligentă. Ca urmare, proiectul *SETI@ONI* își propune să verifice dacă acel semnal provine într-adevăr de la extratereștri sau doar de la niște pușți care beau *Fanta*.

Pentru comoditate, porțiunea de semnal care trebuie analizată vi se pune la dispoziție sub forma unei succesiuni de litere ale alfabetului latin. De asemenea, aveți la dispoziție și un dicționar de cuvinte extraterestre, codificate în același mod. Scopul dumneavoastră este să numărați de câte ori apare fiecare dintre aceste cuvinte în posibilul mesaj extraterestru. Pornind de la aceste date, lingviștii pot să înceapă lucrul la traducerea mesajului.

Date de intrare

Pe prima linie a fișierului de intrare **SETI.IN** se află numărul N al liniilor mesajului. Urmează N linii, fiecare conținând exact 64 de litere ale alfabetului latin, urmate de marcajul de sfârșit de linie. Prin alipirea acestor linii se obține mesajul de analizat, format din $64 \cdot N$ litere.

Pe prima linie a celui de-al doilea fișier de intrare **DIC.IN** este scris numărul M al cuvintelor din dicționar. Urmează M linii, fiecare conținând un cuvânt din dicționar, reprezentat ca o secvență de cel puțin una și cel mult 16 litere. Cuvintele nu sunt neapărat distincte.

Date de ieșire

Fișierul de ieșire **SETI.OUT** va conține exact M linii. Pe linia cu numărul i va fi scris numărul de apariții ale celui de-al i-lea cuvânt din dicționar. Orice apariție a unui cuvânt trebuie numarată, chiar dacă se suprapune peste alte apariții. Se va face distincție între literele mari și literele mici.

Restricții și precizări

- $0 \le N < 2048$;
- $0 \le M \le 32000$;
- un cuvânt poate avea cel mult 65535 de apariții.

Exemplu

SETI.IN

```
2
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaBaba
babaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaBaB
```

DIC.IN

```
3
b
bab
b
```

SETI.OUT

```
3
2
3
```

Timp de execuție: 1 secundă/test

P050216: Suma divizorilor

(Mihai Pătrașcu, Craiova)

Se consideră două numere naturale A și B. Fie S suma tuturor divizorilor naturali ai lui A^B. Să se determine restul împărțirii lui S la 9901.

Date de intrare

Pe prima linie a fișierului de intrare **SUMDIV.IN** se află numerele A și B, separate prin cel puțin un spațiu.

Date de ieșire

Fișierul de ieșire **SUMDIV.OUT** va conține o singură linie pe care se va afla restul împărțirii numărului S la 9901.

Restricție

- $0 \le A, B \le 50.000.000$.

Exemplu

SUMDIV.IN	SUMDIV.OUT
2 3	15

Timp de execuție: 0,5 secunde/test

P050217: Sistem

(Mugurel-Ionuț Andreica, București)

Județul în care are loc *Olimpiada Națională de Informatică* din acest an este special. În județ există N orașe, numerotate de la 1 la N. Fiecare dintre cele N orașe ale județului este legat de ***exact*** alte două orașe, prin străzi bidirecționale. Și mai ciudat este faptul că, în cadrul acestui sistem stradal, nu este întotdeauna posibil să ajungi din orice oraș în oricare alt oraș mergând pe străzi. Oricum, locuitorii județului sunt mândri de acest sistem al lor și sunt de părere că nu mai există altul la fel. Dumneavoastră vreți să le demonstrați contrariul și pentru aceasta vreți să calculați câte sisteme stradale distincte cu proprietatea de mai sus există. Două sisteme sunt considerate distincte dacă există cel puțin o stradă între o pereche de orașe i și j în cadrul primului sistem, care nu exista în cadrul celui de-al doilea.

Date de intrare

Din fișierul **SISTEM.IN** veți citi valoarea N, care reprezintă numărul de orașe ale județului.

Date de ieșire

În fișierul **SISTEM.OUT** se va scrie numărul de sisteme stradale distincte, cu proprietatea că orice oraș este legat prin străzi directe de exact alte două orașe.

Restricție

- $3 \leq N \leq 100$.

Exemple

SISTEM.IN	SISTEM.OUT
4	3

SISTEM.IN	SISTEM.OUT
6	70

Timp de execuție: 1 secundă/test

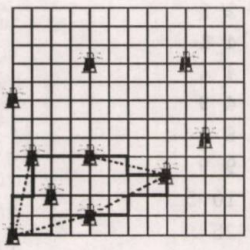
P050218: Comitat

(prof. Rodica Pintea, București)

Toate semințiile conviețuitoare pe *Terra* au hotărât ca *hobbiții*, păstrătorii *Inelului Puterii*, să fie izolați într-o zonă a pământului numită *Comitat*. Hotarele *Comitatului* trebuie să fie reprezentate de un poligon convex cu câte un turn de pază în fiecare vârf.

Se cunosc pozițiile tuturor turnurilor din regiune (două numere naturale raportate la un sistem de axe rectangulare). Un paznic pe cal alb veghează hotarele *Comitatului* parcugând, pe rând, toate distanțele dintre două turnuri succesive mergând pe drum minim, numai pe cărări paralele cu axele.

Se cunoaște lungimea maximă a drumului pe care-l poate parcurge paznicul la un tur complet al hotarelor *Comitatului* și se cere să se determine un poligon cu un număr maxim de turnuri pe contur, poligon care poate constitui hotarul comitatului. În plus, hotarul trebuie să conțină turnul din *Mordor* (de coordonate 0 și 0) într-un vârf, iar în fiecare dintre celelalte vârfuri se află, obligatoriu, unul dintre turnurile existente.



De exemplu, pentru amplasamentul turnurilor ilustrat alăturat și pentru limita de 25 Km a unui tur efectuat de paznic, hotarul *Comitatului* poate fi format, în această ordine, din turnurile de coordonate (0, 0), (4, 1), (8, 3), (4, 4), (1, 4), (0, 0). Se observă că poligonul determinat de aceste turnuri este un poligon convex cu cinci turnuri pe contur.

Poligonul cu vârfurile (0, 0), (4, 1), (4, 12), (0, 7), (0, 0) are tot cinci turnuri pe contur, dar un tur complet al acestui poligon depășește 25 Km.

Date de intrare

Pe prima linie a fișierului de intrare **COMITAT.IN** se află numărul n al turnurilor din ținut (turnul din *Mordor* nu este numărat). Pe fiecare dintre următoarele n linii se află câte două numere naturale, despărțite printr-un spațiu, care reprezintă coordonatele unuia dintre turnuri. Ultima linie a fișierului conține lungimea maximă L a unui tur complet al poligonului.

Date de ieșire

Prima linie a fișierului de ieșire **COMITAT.OUT** va conține numărul v al turnurilor de pe conturul poligonului (incluzând turnul din *Mordor*). Pe a doua linie se vor afla v - 1 numere întregi, reprezentând numerele de ordine ale turnurilor de pe contur, pornind de la turnul din *Mordor* (care nu este afișat) și respectând succesiunea în sens trigonometric sau antitrigonometric a turnurilor de pe contur.

Restricții și precizări

- $0 < n \leq 50$;
- $0 < L < 1000$;
- coordonatele turnurilor sunt numere întregi cuprinse între 0 și 200;
- pot exista turnuri în interiorul poligonului, dar acestea nu sunt luate în considerare pentru numărarea turnurilor corespunzătoare unui poligon;
- există posibilitatea ca soluția să fie dată de un poligon degenerat format dintr-un singur vârf (*Mordor*), din două vârfuri (*Mordor* și un alt turn) sau din mai multe vârfuri coliniare;
- pe conturul poligonului determinat pot exista turnuri coliniare;
- dacă există mai multe soluții care respectă condițiile din enunț, se va furniza doar una dintre acestea;
- în fișierul de intrare nu există două turnuri ale căror poziții să coincidă și singurul turn din poziția (0, 0) este cel din *Mordor*.

Exemplu

COMITAT.IN	COMITAT.OUT
9 0 7 1 4 2 2 4 1 4 4 4 9 8 3 9 9 10 5 25	5 4 7 5 2

Timp de execuție: 1 secundă/test

Barajul de selecție

P050219: EQS

(Mihai Pătrașcu, Craiova)

Se consideră ecuații de forma

$$a_1 \cdot x^3 + a_2 \cdot y^3 + a_3 \cdot z^3 + a_4 \cdot u^3 + a_5 \cdot v^3 = 0.$$

Se consideră soluție un pentuplu (x, y, z, u, v) care verifică ecuația. Să se determine numărul de soluții ale ecuației, pentru care valorile necunoscutelor sunt numere întregi *nenule* din intervalul [-50, 50].

Date de intrare

Fișierul de intrare **EQS.IN** conține o singură linie pe care se află cei cinci coeficienți ai ecuației, separați prin câte un spațiu.

Date de ieșire

Fișierul de ieșire **EQS.OUT** va conține o singură linie pe care se va afla numărul soluțiilor ecuației.

Precizare

- valorile coeficienților, precum și cele ale necunoscutelor, sunt numere întregi cuprinse în intervalul [-50, 50].

Exemplu

EQS.IN	EQS.OUT
37 29 41 43 47	654

Timp de execuție: 2 secunde/test

P050220: Labirint

(Mihai Stroe, București)

Romeo și *Julieta* sunt prinși într-un labirint, descris printr-o matrice cu M linii și N coloane, având elemente din mulțimea {0, 1}. Un element cu valoarea 1 reprezintă un zid, în timp ce un element cu valoarea 0 reprezintă un spațiu liber. *Romeo* se află inițial în colțul din stânga-sus al labirintului (1, 1), iar *Julieta* în colțul din dreapta-jos (M, N).

În pozițiile inițiale ale celor doi nu pot exista ziduri, iar ei se pot deplasa doar în spațiile libere, fără a părăsi matricea.

Pentru a evada din labirint, *Romeo* trebuie să ajungă în poziția (i_1, j_1), iar *Julieta* în poziția (i_2, j_2).

Dumneavoastră dispuneți de un șir de mutări pe care cei doi le pot efectua. Șirul este format din K mutări, codificate prin literele N, E, S și V, reprezentând deplasări de un pătrățel spre *Nord*, *Est*, *Sud*, respectiv *Vest*. *Romeo* și *Julieta* trebuie să efectueze ***toate*** mutările din acest șir, ***în ordinea dată***. Dumneavoastră sunteți cel care decide dacă o anumită mutare va fi efectuată de *Romeo* sau de *Julieta*.

Scrieți un program care va decide care dintre mutări vor fi efectuate de *Romeo* și care de *Julieta* pentru a-i ajuta pe cei doi să ajungă la destinații.

Date de intrare

Fișierul de intrare **LABIRINT.IN** are următoarea structură:

- pe prima linie se află numerele M și N;
- pe următoarele M linii se află descrierea labirintului; fiecare linie conține N valori, separate prin câte un spațiu;
- pe următoarea linie se află patru numere întregi, separate prin câte un spațiu, care reprezintă valorile i_1, j_1, i_2 și j_2;
- pe următoarea linie se află numărul K al mutărilor care trebuie efectuate;
- pe ultima linie se află un șir de K litere care descriu direcțiile în care trebuie efectuate mutările.

Date de ieșire

În fișierul de ieșire **LABIRINT.OUT** se va scrie o singură linie, care conține K caractere. Al i-lea caracter este R, dacă *Romeo* va efectua a i-a mutare sau J dacă mutarea va fi efectuată de *Julieta*.

Restricții și precizări

- $3 \le M \le 20$;
- $3 \le N \le 60$;
- $1 \le K \le 200$;
- există întotdeuna cel puțin o soluție; în cazul în care există mai multe, se va scrie doar una dintre ele;
- prin deplasare la *Nord* se înțelege deplasarea pe linia precedentă, o deplasare la *Sud* înseamnă deplasarea pe linia următoare, deplasarea la *Vest* înseamnă deplasarea pe coloana precedentă, iar deplasarea la *Est* înseamnă deplasarea pe coloana următoare.

Exemplu

LABIRINT.IN	LABIRINT.OUT
5 5 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 2 2 4 4 6 SNEEVV	RJRRJR

Timp de execuție: 1 secundă/test

P050221: ATP

(șef lucrări Stelian Ciurea, Sibiu)

Se știe că jucătorii profesioniști de tenis sunt clasați, în funcție de rezultatele obținute, într-un clasament al *Asociației Tenismanilor Profesioniști* (*ATP*). Studiindu-se rezultatele ultimilor ani, s-a constatat că într-o partidă în care se întâlnesc doi jucători, dacă diferența dintre pozițiile celor doi în clasament este mai mare decât o valoare k, atunci câștigă în mod sigur cel mai bine clasat; în schimb, dacă diferența de poziții în clasament este de cel mult k poziții atunci, teoretic, poate să câștige oricare dintre ei.

La un concurs care urmează să se desfășoare în curând după sistemul eliminatoriu, și-au anunțat participarea primii n jucători din clasament (unde n este o putere a lui 2).

Să se determine care este cel mai slab jucător care ar putea, teoretic, să câștige concursul, precum și schema de desfășurare a partidelor din turneu și câștigătorul fiecărei partide, pentru ca jucătorul respectiv să câștige turneul.

Date de intrare

Fișierului de intrare **ATP.IN** conține o singură linie pe care se află valorile n și k, separate prin spații.

Date de ieșire

Fișierul de ieșire **ATP.OUT** va avea următoarea structură:

- pe prima linie, poziția în clasament a celui mai slab clasat jucător care, teoretic, ar putea câștiga concursul;
- pe a doua linie, întâlnirile din primul tur al turneului; o întâlnire va fi descrisă prin două numere, corespunzătoare pozițiilor din clasament ale jucătorilor care o susțin, iar primul jucător din pereche este câștigătorul; așadar, pe prima linie vor fi $n / 2$ perechi de numere;
- pe a treia linie, întâlnirile din al doilea tur, descrise la fel ca cele din prima linie; pe această linie vor fi $n / 4$ perechi;
- pe ultima linie partida din finală.

Restricție

- $2 \le N \le 4096$.

Exemplu

ATP.IN

```
16 3
```

ATP.OUT

```
11
3 1 5 2 6 4 7 16 8 13 9 14 10 15 11 12
5 3 8 6 9 7 11 10
8 5 11 9
11 8
```

Timp de execuție: 1 secundă/test

P050222: Gard

(Mugurel-Ionuț Andreica, București)

O echipă de K muncitori a fost angajată să vopsească un gard format din N scânduri numerotate de la 1 la N, de la stânga spre dreapta. Fiecare muncitor i se așează în fața scândurii S_i și poate vopsi numai un interval compact (numerele de ordine ale scândurilor din interval sunt consecutive), având maxim L_i scânduri, interval care trebuie să conțină scândura S_i. Pentru fiecare scândură vopsită, acesta este plătit cu suma P_i. Din motive de eficiență, oricare doi muncitori din echipă trebuie să vopsească intervale de scânduri disjuncte.

Fiind conducătorul echipei de muncitori, dumneavoastră doriți să determinați, pentru fiecare membru al echipei, intervalul de scânduri pe care acesta va trebui să îl vopsească, astfel încât câștigul total să fie maxim. Câștigul total este egal cu suma câștigurilor realizate de fiecare membru al echipei. Câștigul realizat de fiecare muncitor este egal cu produsul dintre numărul de scânduri vopsite de acesta și suma P_i corespunzătoare muncitorului.

Scrieți un program care determină câștigul maxim obținut de cei K muncitori.

Date de intrare

Pe prima linie a fișierului de intrare **GARD.IN** se află numărul N al scândurilor și numărul K al muncitorilor, separate printr-un spațiu.

Fiecare dintre următoarele K linii va conține trei numere l, p și s cu semnificația: *muncitorul corepunzător liniei poate vopsi cel mult* l *scânduri, venitul primit pentru fiecare scândură vopsită este* p *și, inițial, el se află în fața scândurii* s.

Date de ieșire

În fișierul de ieșire **GARD.OUT** se va scrie câștigul maxim care poate fi obținut de echipa de muncitori.

Restricție

- $1 \le N \le 16000$;
- $1 \le K \le 100$;
- $1 \le P_i \le 10000$;
- $1 \le L_i, S_i \le N$;
- inițial, doi muncitori nu se pot afla în fața aceleiași scânduri;
- nu trebuie vopsite neapărat toate cele N scânduri ale gardului;
- este posibil ca unul sau mai mulți muncitori să nu vopsească nici o scândură, caz în care scândura în fața căreia s-au așezat inițial poate fi vopsită, eventual, de către alt muncitor.

Exemplu

GARD.IN

```
8 4
3 2 2
3 2 3
3 3 5
1 1 7
```

GARD.OUT

```
17
```

Timp de execuție: 1 secundă/test

Despre concurs...

Ediția din acest an a Olimpiadei Naționale de Informatică a fost un prilej de întâlnire a celor mai talentați elevi din județele țării. Ca de obicei, numărul concurenților a fost foarte mare, la Brăila sosind peste 200 de participanți.

Organizatorii locali s-au achitat excelent de sarcina asumată, din acest punct de vedere neexistând probleme care să afecteze buna desfășurare a competiției.

Din nefericire pentru concurenți, anul acesta numărul premiilor a fost redus. S-au acordat doar trei premii la fiecare clasă, iar următorii clasați au primit mențiuni. În anii anteriori, aproximativ 30% dintre participanți primeau premii, iar alți 20% primeau mențiuni. La ediția din acest an procentul celor care au primit premii sau mențiuni a fost de doar 25%. Cu toate acestea, un aspect pozitiv demn de remarcat este faptul că, acum, un premiu obținut la olimpiada națională are o valoare mult mai mare.

Problemele propuse spre rezolvare au fost destul de dificile, nici unul dintre concurenți nereușind să obțină un punctaj apropiat de cel maxim. Din nefericire, trebuie să remarcăm o continuare a tendinței de scădere a nivelului de pregătire a participanților. Chiar dacă faptul că nu există punctaje apropiate de cel maxim nu reprezintă neapărat un indiciu în acest sens, este îngrijorător faptul că numărul celor care obțin punctaje apropiate de 0 este din ce în ce mai mare.

Punctaje foarte mici apar și la barajul de selecție al lotului național de informatică. Cel de-al cincisprezecelea clasat după cele două probe de baraj a obținut doar 175 de puncte din 600 posibile, deci nu a rezolvat corect decât cel mult una dintre cele șase probleme.

Primii clasați au obținut performanțe mai bune, cel mai mare punctaj obținut la această probă fiind de 473 de puncte.

Situația pare mai îngrijorătoare dacă privim clasamentul pe clase; primul clasat de la clasa a XII-a a obținut doar 305 puncte, adică puțin peste jumătate din punctajul maxim.

În ciuda dificultăților inerente care au apărut, Comisia Centrală și-a făcut din nou datoria, neexistând probleme majore nici din acest punct de vedere.

La această olimpiadă GInfo a fost reprezentată de dl. prof. univ. dr. Horia Georgescu, directorul științific al revistei noastre (președintele Comisiei Centrale) și de d-na lect. univ. Clara Ionescu, redactorul șef al publicației.

P050223: Sistem

(asist. univ. Iuliu Vasilescu, București)

Se consideră N monede identice ca formă și culoare, numerotate de la 0 la N - 1. Printre ele există exact o monedă falsă. Toate monedele adevărate au aceeași greutate, cea falsă fiind mai grea sau mai ușoară decât celelalte. Pentru găsirea monedei false se folosește o balanță cu două talere identice, pe care încap oricâte monede. O cântărire folosind această balanță constă în a pune pe cele două talere ale balanței câte un număr egal de monede. În urma cântăririi se determină dacă greutățile de pe talere sunt egale sau talerul care conține o greutate mai mare.

Scrieți un program care să comande cântăririle care trebuie efectuate, astfel încât să se determine moneda falsă și dacă este mai grea sau mai ușoară decât cele adevărate, în număr minim de cântăriri. Pentru a afla numărul de monede și rezultatul cântăririlor programul va trebui să folosească un modul extern.

Instrucțiuni pentru programatorii în C/C++

Programatorii în *C/C++* au la dispoziție *header*-ul `MON.H`. În acest fișier sunt declarate următoarele tipuri și funcții:

```
typedef char taler[1000];
int init();
int cantarire(taler stanga, taler dreapta);
void rezultat(int moneda, int tip);
```

Instrucțiuni pentru programatorii în Pascal

Programatorii în *Pascal* au la dispoziție *unit*-ul `MON`. În acest *unit* sunt declarate următoarele tipuri, funcții și proceduri:

```
type taler = array[0..999] of Byte;
function init:Integer;
function cantarire(var stanga, dreapta:
                                taler):Integer;
procedure rezultat(moneda, tip:Integer);
```

Instrucțiuni pentru utilizarea modulului

Tipul `taler` este folosit pentru a reprezenta monedele de pe un taler al balanței. Un vector de tipul `taler` va avea elemente din mulțimea {0, 1}. Un vector de acest tip conține pe o poziție valoarea 1, dacă și numai dacă moneda cu numărul corespunzător se află în balanță pe talerul reprezentat prin vector și 0 în caz contrar.

Prima funcție care trebuie apelată este `init`, funcție care returnează numărul de monede.

În continuare va fi apelată, de câte ori este nevoie, funcția `cantarire`, având ca parametri configurațiile celor două talere. Funcția returnează 0 dacă talerele sunt echilibrate, -1 dacă talerul din stânga este mai ușor și 1, dacă talerul din stânga este mai greu.

Procedura/funcția `rezultat` va fi apelată la sfârșit pentru a anunța moneda falsă. Primul parametru este numărul monedei false, iar al doilea trebuie să fie -1 dacă moneda falsă este mai ușoară decât celelalte, respectiv 1 dacă moneda falsă este mai grea decât celelalte.

Nu vor fi realizate operații de intrare/ieșire cu fișiere.

Restricție
• $3 \leq N \leq 1000$.

Timp de execuție: 1 secundă/test

P050224: Banana

(prof. Roxana Tâmplaru și Mihai Pătrașcu, Craiova)

Se consideră o pădure tropicală, reprezentată sub forma unui caroiaj dreptunghiular. Celula din colțul stânga sus al caroiajului are coordonatele (1, 1), iar coordonatele celorlalte celule sunt determinate de linia și coloana pe care se află. În anumite celule ale caroiajului sunt plasați bananieri; o celulă conține cel mult un bananier. Mai mulți bananieri care se învecinează pe orizontală sau verticală formează o zonă de bananieri. Într-o astfel de zonă, *Cekili* se deplasează ușor, cu agilitatea-i cunoscută, de la un bananier la altul.

Maimuța *Cekili* este lacomă și nu îi ajung bananele dintr-o singură zonă. *Tarzan* vrea să-și ajute prietena. Pentru aceasta, el ar putea conecta exact K zone de bananieri înnodând mai multe liane și astfel *Cekili* s-ar putea deplasa de la o zonă la alta utilizând lianele.

Evident, *Tarzan* trebuie să aleagă zonele astfel încât numărul total de bananieri din cele K zone să fie cât mai mare.

Determinați numărul maxim de bananieri care se poate obține prin conectarea a exact K zone.

Date de intrare
Pe prima linie a fișierului de intrare **BANANA.IN** se află numărul Nr al bananierilor și numărul K al zonelor care pot fi conectate, despărțite printr-un spațiu. Pe fiecare dintre următoarele Nr linii se află câte două numere naturale, despărțite printr-un spațiu, care reprezintă coordonatele unuia dintre bananieri.

Date de ieșire
Fișierul de ieșire **BANANA.OUT** va conține o singură linie pe care se va afla numărul maxim de bananieri care se poate obține prin conectarea zonelor.

Restricții și precizări
• $1 \leq Nr \leq 16000$;
• coordonatele turnurilor sunt numere întregi cuprinse între 1 și 10000;
• numărul de zone este cel puțin egal cu K;
• două poziții se învecinează pe orizontală dacă sunt pe aceeași linie și pe coloane consecutive, respectiv pe verticală dacă sunt pe aceeași coloană și pe linii consecutive.

Exemplu

BANANA.IN	BANANA.OUT
10 3 7 10 1 1 101 1 2 2 102 1 7 11 200 202 2 1 3 2 103 1	9

Timp de execuție: 1 secundă/test

Premianții

Clasa a IX-a
Leonard Crestez, Brăila - premiul I
Iolanda Popa, Iași - premiul II
Ionuț Costică, Iași - premiul III

Clasa a X-a
George-Dan Ghinea, București - premiul I
Andrei Homescu, Gorj - premiul II
Cristian Cutocheraș, Dâmbovița - premiul III

Clasa a XI-a
Radu Berinde, București - premiul I
Victor Costan, București - premiul II
Cosmin Răianu, Constanța - premiul III

Clasa a XII-a
Daniel Dumitran, București - premiul I
Vlad Dascălu, Bacău - premiul II
Ștefăniță Fechete, Suceava - premiul III

Membrii lotului național
Andrei Benea, Vrancea
Radu Berinde, București
Ștefan Ciobâcă, Suceava
Victor Costan, București
Vlad Dascălu, Bacău
Alexandru Dumitrache, Argeș
Daniel Dumitran, București
George-Dan Ghinea, București
Claudiu Gruia, București
Bogdan Hârjoc, Alba
Andrei Markovits, Satu Mare
Paul Marinescu, Buzău
Cosmin Răianu, Constanța
Cristian-Francisc Toth, Timiș
Victor Vernescu, Constanța

Craiova, județul Dolj

Concursul de informatică ȘTEFAN ODOBLEJA

În perioada 5-7 martie 2002 s-a desfășurat la Craiova un concurs interjudețean de programare la care au participat elevi din nouă județe ale țării. În continuare vă prezentăm enunțurile celor zece probleme cu care s-au confruntat concurenții.

Clasa a IX-a

P050225: Puzzle

(lect. univ. dr. Petre Băzăvan)

Se consideră o imagine alcătuită din mai multe piese de formă dreptunghiulară, fiecare având aceleași dimensiuni. Imaginea este formată din n linii și n coloane, fiecare element al acesteia fiind o piesă. Elementele componente ale unei piese sunt simboluri din setul de caractere *ASCII*. Sarcina voastră este de a reconstitui imaginea inițială pe baza pieselor componente. Fiecare piesă este caracterizată prin patru numere, corespunzătoare celor patru laturi ale sale. Dacă unul dintre aceste numere este 0, înseamnă că latura respectivă se află pe marginea imaginii inițiale. Dacă valoarea unui număr este x ($x \neq 0$), atunci latura respectivă se va învecina cu latura unei alte piese căreia îi corespunde numărul $-x$. Piesele nu pot fi rotite și sunt unic determinate de cele patru numere care descriu marginile.

Date de intrare

Fișierul de intrare **PUZZLE.IN** are următoarea structură:

- prima linie conține trei numere naturale n, h, l, unde n reprezintă dimensiunea imaginii (numărul de piese) care trebuie reconstituită, iar numerele h și l reprezintă înălțimea, respectiv lățimea unei piese;
- următoarele linii conțin descrierea celor n^2 piese; acestea vor fi separate prin câte o linie vidă; fiecare piesă va fi descrisă prin $h + 1$ linii astfel:
  - ♦ primele h linii vor conține câte l caractere care descriu liniile piesei;
  - ♦ următoarea linie va conține cele patru numere care descriu marginile piesei; numerele corespunzătoare laturilor apar în ordinea *sus*, *stânga*, *jos*, *dreapta*.

Date de ieșire

Fișierul de ieșire **PUZZLE.OUT** va conține $n \cdot h$ linii, pe fiecare dintre acestea aflându-se $n \cdot l$ caractere. Aceste caractere vor descrie imaginea reconstituită.

Restricții și precizări

- $2 \leq n \leq 10$;
- $1 \leq h, l \leq 25$;
- laturile pieselor sunt caracterizate prin numere întregi cuprinse între -5 și 5;
- întotdeauna există o singură soluție;
- spațiile sunt caractere valide care pot intra în componența pieselor.

Exemplu

PUZZLE.IN

```
2 8 14
88,
8888.
:8888b
  8888
-.:888b
' d8888
 ,88888
':88888
0 3 -1 0

        o8%88
      o88%888
     8'-     -
    8'
   d8.-=. ,==
   >8 `~` :`~
   88
   88b. `--
0 0 -5 -3

.:88888
:::8888
:' 8888b
    8888b
    ,%888b.
    %%%8--'-.
   _%-' ---  -
.-'   =  --.
1 4 0 0

    888b ~==~
    88888o--:'
    `88888| ::
    8888^^'
   d888
  d88%
 /88:.__ ,
     '''::===.
5 0 0 -4
```

PUZZLE.OUT

```
        o8%8888,
      o88%8888888.
     8'-     -:8888b
    8'          8888
   d8.-=. ,==-.:888b
   >8 `~` :`~' d8888
   88          ,88888
   88b. `--  ':88888
   888b ~==~ .:88888
   88888o--:':::8888
   `88888| :::' 8888b
   8888^^'        8888b
  d888            ,%888b.
 d88%            %%%8--'-.
/88:.__ ,       _%-' ---  -
     '''::===..-'   =  --.
```

Timp de execuție: 1 secundă/test

Despre concurs...

Concursul de programare "Ştefan Odobleja" a fost organizat de Liceul de Informatică din Craiova, în colaborare cu Inspectoratul Şcolar Judeţean Gorj şi Facultatea de Matematică -Informatică din cadrul Universităţii din Craiova.

La concurs au participat 44 de elevi din nouă judeţe ale ţării: Alba, Braşov, Cluj, Dolj, Gorj, Iaşi, Mehedinţi, Olt şi Vâlcea.

Au fost două secţiuni: una la care concurenţii au participat individual şi una la care reprezentanţii fiecărui judeţ, indiferent de clasă, au format echipe.

Redacţia GInfo a fost reprezentată la acest concurs de redactorul şef Clara Ionescu, precum şi de redactorul Eugen Ionescu.

P040226: Ecuaţii

(asist. univ. Florin Stavu)

Scrieţi un program care rezolvă ecuaţii de gradul întâi. Ecuaţiile sunt descrise potrivit următoarei sintaxe:

- *ecuaţie*: *expresie* = *expresie*;
- *expresie*: *termen* **sau** *expresie* (+ **sau** -) *termen*;
- *termen*: *factor* **sau** *factor* * *factor*;
- *factor*: *număr* **sau** x **sau** (*expresie*);
- *număr*: *cifră* **sau** *cifră număr*;
- *cifră*: 0 **sau** 1 **sau** ... **sau** 9.

Conform acestei sintaxe nu se pot scrie ecuaţii care să conţină termeni de forma x*x. Se garantează că fişierul de intrare va conţine doar ecuaţii liniare în *x* (ecuaţii de gradul întâi).

De asemenea, toate subexpresiile unei ecuaţii vor fi liniare în *x* (de exemplu, nu poate apărea expresia x*x-x*x+3*x=9).

Date de intrare

Fişierul de intrare **ECUATII.IN** conţine mai multe ecuaţii, câte una pe fiecare linie.

Date de ieşire

În fişierul de ieşire **ECUATII.OUT** se vor scrie rezultatele ecuaţiilor. Pentru fiecare ecuaţie se vor scrie două linii: prima dintre ele va conţine textul Ecuatia #i, unde i este numărul de ordine al ecuaţiei; a doua linie va conţine unul dintre următoarele mesaje:

- Nu are solutie, în cazul în care ecuaţia nu are soluţii;
- O infinitate de soluţii, în cazul în care ecuaţia are o infinitate de soluţii;
- x=valoare, unde valoare este numărul real (tipărit cu şase cifre zecimale) care reprezintă valoarea soluţiei, în cazul în care există o singură soluţie a ecuaţiei.

Restricţie

- fiecare ecuaţie este descrisă prin cel mult 100 de caractere.

Exemplu

ECUATII.IN	ECUATII.OUT
x+x=10	Ecuatia #1
3*x+2=21	x = 5.000000
3*x=3*x+1+2+3	Ecuatia #2
(12-3*4)*x=2*5-10	x = 6.333333
	Ecuatia #3
	Nu are solutie
	Ecuatia #4
	O infinitate de solutii

Timp de execuţie: 1 secundă/test

Clasa a X-a

P040227: Arheolog

(asist. univ. Mihai Stancu)

Un arheolog care cerceta vestigiile unei civilizaţii dispărute cu mii de ani în urmă, a găsit şapte plăcuţe hexagonale pe care erau desenate diverse inscripţii. Dorind să descifreze mesajul, el descoperă că plăcuţele trebuie alipite într-un anume fel. Arheologul aşează cele şapte plăcuţe (identificate prin literele A, B, C, D, E, F şi G) pe şapte tije verticale (identificate prin numerele 0, 1, 2, 3, 4, 5 şi 6; tija din centru este identificată prin 0). În figura 1 este ilustrat modul de amplasare a tijelor.

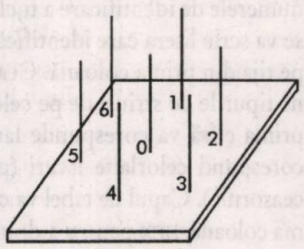


Fig. 1: Tijele

Două plăcuţe hexagonale se pot îmbina dacă au acelaşi tip de striuri pe laturile care vor fi învecinate. Există exact zece tipuri de striuri, identificate prin cele zece cifre. În figura 2 este ilustrată plăcuţa D care are laturi cu diferite tipuri de striuri.

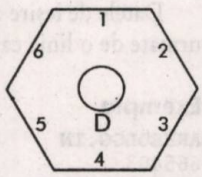


Fig. 2: Plăcuţă

Cele şapte hexagoane sunt amplasate pe cele şapte tije, fiecare hexagon din margine având trei laturi lipite de laturile altor hexagoane, în timp ce hexagonul de pe tija 0 are toate laturile lipite de cele ale altor hexagoane. Pentru a ajuta la rezolvarea misterului, va trebui să găsiţi un aranjament şi o orientare pentru plăcuţele hexagonale, astfel încât striurile de pe oricare două laturi lipite să fie identice. O configuraţie corectă este ilustrată în figura 3.

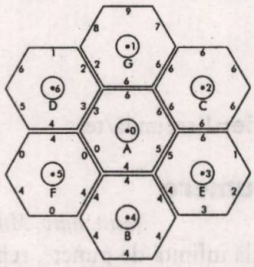


Fig. 3: Amplasament corect

Date de intrare

Fișierul **ARHEOLOG.IN** conține mai multe seturi de date corespunzătoare mai multor descrieri ale plăcuțelor. Fiecare set de plăcuțe va fi descris prin opt linii. Primele șapte linii conțin câte șase cifre (neseparate prin spații) care indică tipurile de striuri de pe laturile plăcuțelor. Prima linie conține cifrele corespunzătoare hexagonului A, a doua linie conține cifrele corespunzătoare hexagonul B etc. Prima cifră de pe fiecare linie corespunde laturii de sus, iar următoarele corespund celorlalte laturi (parcurse în sensul acelor de ceasornic). De exemplu, hexagonul din figura 1 este reprezentat prin secvența de cifre 123456. Cea de-a opta linie va conține doar caracterul * și are rolul de a delimita seturile de date.

Date de ieșire

Fișierul de ieșire **ARHEOLOG.OUT** va conține câte un tabel pentru fiecare set de date. Acesta va fi format dintr-un cap de tabel și șapte linii a câte trei coloane. Prima coloană corespunde tijelor, cea de-a doua hexagoanelor, iar cea de-a treia orientării plăcuțelor. În prima coloană se vor scrie numerele de identificare a tijelor. Pe cea de-a doua coloană se va scrie litera care identifică plăcuța care va fi amplasată pe tija din prima coloană. Cea de-a treia coloană va conține tipurile de striuri de pe cele șase laturi ale hexagonului; prima cifră va corespunde laturii de sus, iar următoarele corespund celorlalte laturi (parcurse în sensul acelor de ceasornic). Capul de tabel va conține textul TIJ pentru prima coloană, HEX pentru a doua și POZITIE pentru a treia.

În cazul în care nu există nici o soluție, se va scrie doar mesajul `Nu exista solutii.`

Datele de ieșire corespunzătoare unui set de date vor fi urmate de o linie care va conține 20 de caractere *.

Exemplu

ARHEOLOG.IN

```
665403
444444
666666
123456
123456
040404
289766
*
123456
111111
222222
333333
444444
555555
666666
*
```

ARHEOLOG.OUT

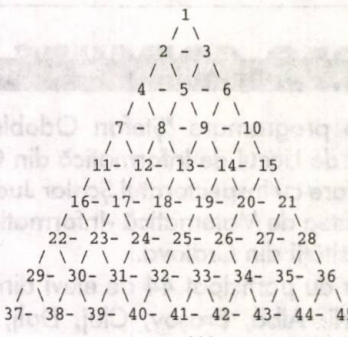
```
TIJ    HEX    POZITIE
 0      A     665403
 1      G     976628
 2      C     666666
 3      E     612345
 4      B     444444
 5      F     404040
 6      D     123456
********************
Nu exista solutii.
********************
```

Timp de execuție: 1 secundă/test

P030228: Numere

(asist. univ. Mihai Gabroveanu)

Se consideră o grilă infinită de puncte, reprezentând vârfuri ale unor triunghiuri echilaterale așezate astfel:



Aceste puncte sunt numerotate de la stânga la dreapta și de sus în jos. Grupuri de puncte pot forma laturile unor figuri geometrice. De exemplu, mulțimile de puncte {1,2,3} și {7,9,18} reprezintă vârfurile unor triunghiuri, mulțimile {11,13,26,24} și {2,7,9,18} reprezintă vârfurile unor paralelograme, iar mulțimile {4,5,9,13,12,7} și {8,10,17,21,32,34} reprezintă vârfurile unor hexagoane.

Va trebui să scrieți un program care stabilește dacă o mulțime de puncte de pe această grilă reprezintă laturile unei figuri acceptate: *triunghi*, *paralelogram*, *hexagon*.

O figură acceptată trebuie să îndeplinească următoarele condiții:

- fiecare latură a figurii trebuie să coincidă cu o muchie din grilă (prin muchie se înțelege unul, două sau mai multe segmente care unesc puncte adiacente coliniare);
- laturile figurii trebuie să aibă lungimi egale.

Date de intrare

Intrarea va consta dintr-un număr necunoscut de mulțimi de puncte. Fiecare mulțime de puncte va apărea pe câte o linie a fișierului **NUMERE.IN** . Fiecare mulțime conține cel mult șase puncte, iar numerele corespunzătoare acestora sunt cuprinse între 1 și 32767.

Date de ieșire

Fișierul de ieșire **NUMERE.OUT** va conține câte o linie pentru fiecare mulțime din fișierul de intrare. Pe această linie se va scrie un mesaj care va descrie tipul figurii reprezentate de punctele mulțimii: TRIUNGHI, PARALELOGRAM sau HEXAGON.

În cazul în care figura corespunzătoare nu este acceptată, linia va conține mesajul NU.

Exemplu

NUMERE.IN

```
1 2 3
11 13 29 31
26 11 13 24
4 5 9 13 12 7
1 2 3 4 5
47
11 13 23 25
```

NUMERE.OUT

```
TRIUNGHI
NU
PARALELOGRAM
HEXAGON
NU
NU
NU
```

Timp de execuție: 1 secundă/test

Clasele a XI-a și a XII-a

P030229: Partiție

(asist. univ. Mirel Coșulschi, asist. univ. Nicolae Constantinescu)

Fie un număr natural M; numim partiție a lui M două secvențe de numere întregi $x_1, x_2, ..., x_k$ și $n_1, n_2, ..., n_k$, astfel încât:

- $x_1 < x_2 < ... < x_k$;
- $n_1 \cdot x_1 + n_2 \cdot x_2 + ... + n_k \cdot x_k = M$.

Prin sumă parțială a unei partiții se înțelege expresia $m_1 \cdot x_1 + m_2 \cdot x_2 + ... + m_k \cdot x_k$, unde $m_i \le n_i$.

Se cere să se determine o partiție a numărului M care îndeplinește următoarele condiții:

- orice număr cuprins între 1 și M poate fi reprezentată în mod unic ca o sumă parțială a acestei partiții;
- partiția conține un număr maxim de elemente distincte.

De exemplu, pentru $M = 7$, câteva dintre posibilitățile de partiționare sunt:

- $7 = 7 \cdot 1$ ($x_1 = 1, n_1 = 7$);
- $7 = 1 \cdot 1 + 3 \cdot 2$ ($x_1 = 1, x_2 = 2, n_1 = 1, n_2 = 3$);
- $7 = 1 \cdot 1 + 1 \cdot 2 + 1 \cdot 4$ ($x_1 = 1, x_2 = 2, x_3 = 4, n_1 = 1, n_2 = 1, n_3 = 1$);
- $7 = 3 \cdot 1 + 2 \cdot 2$ ($x_1 = 1, x_2 = 2, n_1 = 3, n_2 = 2$).

Ultima dintre aceste partiții nu îndeplinește prima condiție, deoarece numărul 2 poate fi scris în două moduri ca sumă parțială a partiției:

- $2 = 2 \cdot 1 + 0 \cdot 2$ ($m_1 = 2, m_2 = 0$);
- $2 = 0 \cdot 1 + 1 \cdot 2$ ($m_1 = 0, m_2 = 1$).

Celelalte partiții îndeplinesc această condiție; ca urmare, partiția căutată este $1 \cdot 1 + 1 \cdot 2 + 1 \cdot 4$, deoarece conține mai multe elemente decât celelalte.

Date de intrare

Fișierul de intrare **PARTITIE.IN** conține o singură linie pe care se află valoarea M.

Date de ieșire

Fișierul de ieșire **PARTITIE.OUT** va conține trei linii. Pe prima dintre ele se va afla valoarea k care indică numărul de elemente ale partiției.

Cea de-a doua linie va conține valorile $n_1, n_2, ..., n_k$, separate prin câte un spațiu.

Ultima linie va conține valorile $x_1, x_2, ..., x_k$, separate prin câte un spațiu.

Restricție

- $1 \le M \le 255$;

Exemplu

PARTITIE.IN	PARTITIE.OUT
7	3 1 1 1 1 2 4

Timp de execuție: 1 secundă/test

P030230: Bile

(asist. univ. Mirel Coșulschi, asist. univ. Nicolae Constantinescu)

Un distribuitor de bile este un aparat cu o intrare și două ieșiri (*stângă* și *dreaptă*) care funcționează astfel: la intrare primește un șir de bile pe care le distribuie alternativ prin cele două ieșiri: prima bilă iese prin stânga, a doua prin dreapta, a treia prin stânga etc.

Să considerăm acum un arbore format din astfel de distribuitoare. Într-un astfel de arbore bilele se introduc prin vârf și trec succesiv prin distribuitoare în funcție de starea fiecărui distribuitor, până ajung la o ieșire care nu este conectată la nici un distribuitor; spunem că în acest moment bila *a părăsit arborele*.

Dacă ieșirea prin care bila părăsește arborele este ieșirea din stânga a unui distribuitor, atunci se spune că bila *a părăsit arborele prin stânga*; analog, dacă ieșirea prin care bila părăsește arborele este ieșirea din dreapta a unui distribuitor, atunci se spune că bila *a părăsit arborele prin dreapta*.

Vom considera un aparat format dintr-un astfel de arbore sub care se află o tăviță cu N bile, numerotate de la 1 la N, așezate inițial în ordine crescătoare de la stânga la dreapta.

La fiecare pas toate bilele sunt introduse în arbore, în ordinea în care se aflau în tăviță, de la stânga la dreapta. Fiecare bilă care va părăsi arborele prin stânga se va așeza în stânga bilelor prezente în tăviță în momentul căderii ei și, analog, fiecare bilă care va părăsi arborele prin dreapta se va așeza în dreapta bilelor din tăviță.

O bilă este introdusă în arbore doar după ce predecesoarea sa a ajuns deja în tăviță. După fiecare astfel de pas, toate bilele se vor afla în tăviță, dar în altă ordine.

Pentru o configurație dată, să se determine numărul minim de pași după care bilele ajung în tăviță în ordinea inițială.

Date de intrare

Fișierul de intrare **BILE.IN** are următoarea structură:

- pe prima linie se află două numere întregi pozitive N și M care reprezintă numărul bilelor, respectiv cel al distribuitoarelor;
- pe următoarele M linii se află câte trei numere întregi pozitive separate printr-un singur spațiu i, j, k care indică faptul că distribuitorul j este conectat la ieșirea stângă a distribuitorului i, iar distribuitorul k este conectat la ieșirea dreaptă a distribuitorului i; în cazul în care una dintre valorile j sau k este 0, atunci ieșirea corespunzătoare nu este conectată la nici un distribuitor.

Date de ieșire

Fișierul de ieșire **BILE.OUT** va conține o singură linie pe care se va afla numărul minim de pași necesari pentru ca bilele să se afle în tăviță în ordinea inițială.

Restricții și precizări

- $0 < N, M \le 1000$;

- în vârf se află întotdeauna distribuitorul cu numărul de ordine 1;
- după fiecare pas distribuitoarele sunt "resetate", astfel încât prima bila pe care o vor primi la pasul următor o vor distribui prin stânga.

Exemplu

BILE.IN	BILE.OUT
4 4	2
1 2 3	
2 0 0	
3 4 0	
4 0 0	

Timp de execuție: 1 secundă/test

P030231: For

(asist. univ. Mirel Coșulschi)

Fie următorul fragment de program scris în limbajul *Pascal*:

```
...
for i:=1 to 20 do
    for j := 1 to i do
        write('*');
...
```

Se poate verifica foarte ușor faptul că instrucțiunea formată din apelul procedurii `write` se va executa de 210 ori.

Se dau mai multe instrucțiuni de ciclare **`for`** imbricate, corpul ultimei instrucțiuni de ciclare conținând un apel `write`.

Se cere să se scrie un program care să determine de câte ori se va apela procedura `write` în timpul execuției fragmentului respectiv.

Date de intrare

Fișierul de intrare **`FOR.IN`** va conține pe prima linie numărul n al instrucțiunilor de ciclare imbricate. Pe fiecare dintre următoarele n linii se va afla un triplet de forma *var*, exp_1, exp_2; *var* este un identificator valid în limbajul *Pascal* și reprezintă variabila de ciclare pentru o instrucțiune **`for`**; exp_1 și exp_2 pot fi constante sau identificatorul variabilei de ciclare corespunzatoare instrucțiunii **`for`** de pe nivelul anterior.

Date de ieșire

Fișierul de ieșire **`FOR.OUT`** va conține o singură linie pe care se va afla numărul de execuții al instrucțiunii care conține apelul procedurii `write`.

Restricții și precizări

- $0 < n < 20$;
- constantele prezente în fișierul de intrare sunt numere naturale mai mici decât 1000.

Exemplu

FOR.IN	FOR.OUT
2	210
i 1 20	
j 1 i	

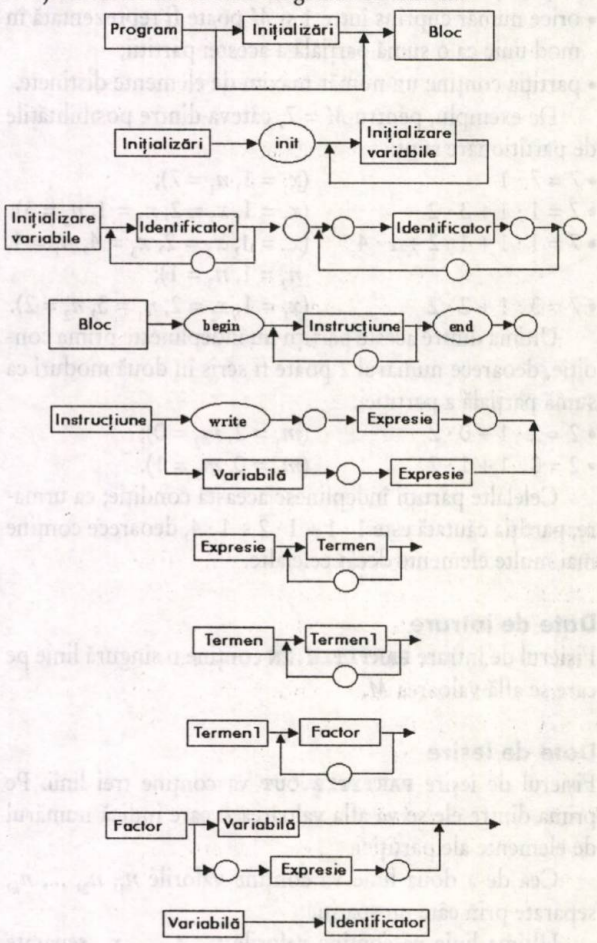
Timp de execuție: 1 secundă/test

Concursul pe echipe

P030232: MultiX

(asist. univ. Mirel Coșulschi)

Cosiderăm un nou limbaj de programare pe care îl definim cu ajutorul următoarelor diagrame de sintaxă:



Identificatorii sunt definiți într-un mod asemănător cu cel din *Pascal*: orice combinație de litere și cifre, care începe obligatoriu cu o literă și are lungimea mai mică decât 255 de caractere. O variabilă identifică o mulțime.

Se definesc următoarele operații:

- `*` - intersecția a două mulțimi;
- `+` - reuniunea a două mulțimi;
- `*` - diferența a două mulțimi;

Numim acest limbaj *MultiX*. Fie următorul program scris în acest limbaj:

```
init a : "a", "b", "c";
     b,c : "a", "c", "d";
```

```
begin
  a := a + b;
  write(a);
  b := a - (a * b);
  write(b);
end.
```

Interpretarea acestui program este următoarea:

- se inițializează mulțimile A = {'a', 'b', 'c'}, B = {'a', 'c', 'd'} și C = {'a', 'c', 'd'};
- prin instrucțiunea `a := a + b` se calculează reuniunea mulțimilor A și B, iar rezultatul este atribuit variabilei care reprezintă mulțimea A; această mulțime devine A = {'a', 'b', 'c', 'd'};
- prin instrucțiunea `write(a)` se afișează elementele mulțimii A; așadar, se va afișa `a b c d`;
- prin instrucțiunea `b := a - (a * b)` se calculează diferența dintre mulțimea A și intersecția mulțimilor A și B, iar rezultatul este atribuit variabilei care reprezintă mulțimea B; această mulțime devine B = {'b'};
- prin instrucțiunea `write(b)` se afișează elementele mulțimii B; așadar, se va afișa `b`.

Să se realizeze o aplicație care, pentru un program scris în limbajul *MultiX*, îi validează corectitudinea conform diagramelor de sintaxă anterioare și, în cazul unui program valid, va afișa rezultatul execuției acestuia.

Date de intrare

Fișierul de intrare **MULTIX.IN** va conține un program scris în limbajul *MultiX*.

Date de ieșire

Dacă programul conține erori sintactice, fișierul de ieșire **MULTIX.OUT** va conține mesajul `Eroare sintactica pe linia x`, unde `x` este numărul liniei pe care a apărut prima eroare. În cazul în care nu există erori sintactice, fișierul va conține rezultatele afișate de program. Fiecare linie a fișierului va corespunde unui apel `write`; elementele mulțimilor vor fi separate prin câte un spațiu.

Restricții și precizări

- numărul variabilelor prezente în program este de cel mult 255;
- cardinalul unei mulțimi nu va depăși valoarea 1000;
- fiecare linie a programului conține o singură instrucțiune;
- o variabilă care nu apare în secțiunea de inițializare se consideră a fi inițializată cu mulțimea vidă;
- limbajul *MultiX* este *case sensitive*: se fac deosebiri între litere mici și litere mari;
- un apel de scriere, al cărui argument este mulțimea vidă, va duce la tipărirea simbolului $ pe linia corespunzătoare;
- o linie a programului poate conține cel mult 1000 de caractere.

Exemplu

MULTIX.IN	MULTIX.OUT

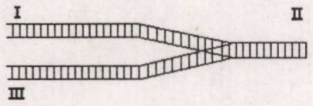
`init x: "a1","a2","a3";`
`y,z: "u","a2","z";`
`begin`
`write(x * y);`
`z := z + (y - x);`
`write(z);`
`end.`
	`a2`
	`u a2 z`

Timp de execuție: 1 secundă/test

P030233: Depou

(asist. univ. Mirel Coșulschi)

Într-un depou cu trei linii, suficient de lungi, avem o garnitură de tren formată din N vagoane, numerotate de la 1 la N. Inițial vagoanele se află pe linia I și garnitura trebuie să ajungă pe linia III.



Un vagon se poate deplasa de pe linia I pe linia II (dacă nu există nici un vagon la dreapta sa) sau de pe linia II pe linia III (dacă nu există nici un vagon la stânga sa). Vagoanele nu se pot deplasa în sens invers (de pe linia II pe linia I sau de pe linia III pe linia II) sau direct de pe linia I pe linia III. La fiecare pas se poate deplasa de pe o linie pe alta un singur vagon. Să considerăm trenul format din vagoanele 1, 2 și 3.

Premianții - concursul individual

Clasa a IX-a

Andrei Ioniță, Olt - premiul I
Tudor-Alexandru Lițu, Iași - premiul II
Alexandru Crețu, Dolj - premiul III
Ciprian Stănescu, Dolj - premiul III

Clasa a X-a

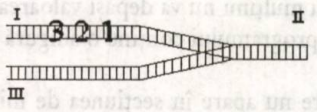
Flaviu Roman, Cluj - premiul I
Radu Pârvu, Brașov - premiul II
Iolanda Popa, Iași - premiul III

Clasa a XI-a

Alexandru Surpățean, Brașov - premiul I
Horia-Andrei Ciochină, Iași - premiul II
Cezar Stalea, Dolj - premiul III
Marius Truică, Dolj - premiul III

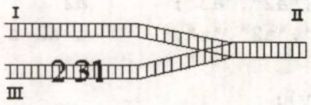
Clasa a XII-a

Bogdan Hârjoc, Alba - premiul I
Mihai Iacob, Brașov - premiul II
Cristian Ivănescu, Dolj - premiul III



O posibilitate de mutare a vagoanelor de pe linia I pe linia III ar putea fi:
- pasul 1: vagonul 1 trece de pe linia I pe linia II;
- pasul 2: vagonul 2 trece de pe linia I pe linia II;
- pasul 3: vagonul 2 trece de pe linia II pe linia III;
- pasul 4: vagonul 3 trece de pe linia I pe linia II;
- pasul 5: vagonul 3 trece de pe linia II pe linia III;
- pasul 6: vagonul 1 trece de pe linia II pe linia III;

După efectuarea acestor pași, vagoanele ajung pe linia III în ordinea 2, 3, 1.



Să se scrie un program care determină numărul variantelor în care pot fi așezate vagoanele pe linia III.

Date de intrare
Fișierul de intrare **DEPOU.IN** va conține numărul vagoanelor de pe linia I a depoului.

Date de ieșire
Fișierul de ieșire **DEPOU.OUT** va conține numărul configurațiilor posibile în care se pot afla vagoanele pe linia III a depoului.

Restricție
- $1 \le N \le 100$.

Exemplu

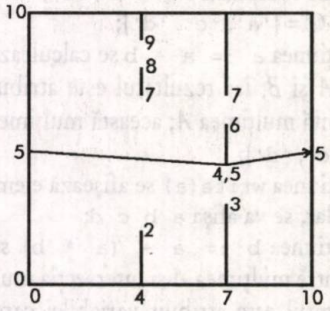
DEPOU.IN	DEPOU.OUT
3	5

Timp de execuție: 1 secundă/test

P030234: Cameră

(asist. univ. Mihai Gabroveanu)

Să se determine lungimea celui mai scurt drum printr-o cameră ce conține pereți despărțitori. Camera va avea întotdeauna ca margini pereții delimitați de punctele de coordonate (0, 0), (10, 0), (10, 10), (0, 10), după cum se poate vedea în figura de mai jos. Punctul de plecare este întotdeauna (0, 5), iar punctul de sosire este (10, 5). În cameră pot exista k pereți despărțitori, fiecare dintre ei având două uși. Figura prezintă o cameră cu doi pereți despărțitori; este ilustrat și drumul minim corespunzător camerei respective.



Date de intrare
Fișierul de intrare **CAMERA.IN** conține pe prima linie numărul k al pereților despărțitori. Fiecare dintre următoarele k linii conține descrierea unui perete. O linie va conține cinci numere reale; primul dintre ele reprezintă coordonata orizontală a peretelui. Următoarele două numere reprezintă coordonatele verticale ale primei uși, în timp ce ultimele două corespund coordonatelor verticale ale celei de a doua uși.

Date de ieșire
Fișierul de ieșire **CAMERA.OUT** va conține o singură linie pe care se va afla un număr real (scris cu două zecimale exacte) care reprezintă lungimea celui mai scurt drum.

Restricție
- $0 \le k \le 18$.

Exemplu

CAMERA.IN	CAMERA.OUT
2 4 2 7 8 9 7 3 4.5 6 7	10.06

Timp de execuție: 1 secundă/test

Premianții - concursul pe echipe

Premiul I
Andrei Homescu, Emanuel Udrea, Silviu Udrea - Gorj

Premiul II
Cristian Domșa, Bogdan Hârjoc, Ionuț Turus - Alba

Premiul III
Mircea Ispas, Constantin Jianu, Lucian Morăreț, Mihai Udrea - Dolj

Centrul de excelență, Iași

Concursul de informatică
URMAȘII LUI MOISIL

În perioada 18-20 martie 2002 s-a desfășurat la Iași concursul de informatică "Urmașii lui Moisil". Vă prezentăm în continuare enunțurile celor șase probleme propuse spre rezolvare la acest concurs.

Clasa a IX-a

P050235: Roboței

(prof. Anamaria Rusu, prof. Tudor Boutiuc)

Pe o platformă dreptunghiulară situată în *Oceanul Atlantic*, în perioada marilor furtuni lucrează numai roboți. Harta platformei este o matrice cu $M + 1$ linii și $N + 1$ coloane, numerotate de la 0 la M, respectiv de la 0 la N. Pe platformă există R roboți, numerotați distinct de la 1 la R. Pentru fiecare robot se cunoaște poziția inițială și o secvență de comenzi care determină modul de deplasare a robotului pe platformă.

Poziția unui robot este specificată printr-o pereche de coordonate întregi de pe hartă și o orientare (N - nord, E - est, S - sud, V - vest).

Direcția N corespunde deplasării din punctul (x, y) în punctul $(x, y + 1)$.

O secvență de comenzi este un șir de litere S, D, I, cu semnificația:

- S - stânga: robotul se întoarce la stânga cu 90° și rămâne în același punct;
- D - dreapta: robotul se întoarce la dreapta cu 90° și rămâne în același punct;
- I - înainte: robotul se deplasează cu un punct în direcția în care este orientat, păstrându-și orientarea.

Un robot care se deplasează în afara platformei este pierdut pentru totdeauna. Totuși, un robot pierdut lasă un mesaj care va împiedica alți roboți să cadă de pe platformă din același punct. Mesajul este lăsat în ultimul punct ocupat de robot înainte de dispariție. O comandă de deplasare în afara platformei din acel punct va fi ignorată de ceilalți roboți.

Roboții lucrează secvențial, adică roboții urcă pe platformă pe rând, conform numerelor lor de ordine, se deplasează în poziția finală, apoi coboară de pe platformă.

Sarcina voastră este de a realiza un program care determină poziția finală de pe platformă a fiecărui robot.

Date de intrare

Fișierul de intrare **ROBO.IN** are următoarea structură:

- prima linie conține două numere naturale M și N, care reprezintă coordonatele colțului dreapta-sus ale hărții (colțul stânga-jos are coordonatele 0, 0);
- a doua linie conține un număr natural R care reprezintă numărul de roboți;
- următoarele $2 \cdot R$ linii sunt împărțite în R grupuri și pe prima linie din fiecare grup se află două numere naturale, separate printr-un singur spațiu, care reprezintă coordonatele la care se află un robot și un caracter care reprezintă orientarea robotului, iar a doua linie conține o secvență de comenzi.

Date de ieșire

Fișierul text **ROBO.OUT** va avea R linii și pe fiecare linie se va scrie poziția finală a unui robot și orientarea sa. Dacă un robot cade de pe platformă se va scrie mesajul PIERDUT după ultima lui poziție pe platformă. Pe fiecare linie datele vor fi separate printr-un singur spațiu.

Restricții și precizări

- $1 \le M, N \le 50$;
- $1 < R < 100$;
- secvențele de comenzi conțin cel mult 100 de caractere;
- toate pozițiile inițiale sunt în interiorul platformei.

Exemplu

ROBO.IN

```
5 3
3
1 1 E
DIDIDIDI
3 2 N
IDDISSIIDDISS
0 3 V
SSIIISISIS
```

ROBO.OUT

```
1 1 E
3 3 N PIERDUT
2 3 S
```

Timp de execuție: 1 secundă/test

P050236: T9

(prof. Marinel Șerban)

T9 este un sistem dezvoltat pentru a satisface nevoia de a trimite cât mai rapid mesaje text (*SMS*) utilizând telefoane mobile. Un mesaj text constă dintr-o succesiune de cuvinte separate prin câte un spațiu.

Sistemul T9 este bazat pe un dicționar stocat în memoria telefonului. Dacă utilizăm acest sistem, atunci când tastăm un cuvânt este suficient să apăsăm pentru fiecare literă o singură dată o tastă. La acționarea unei taste este afișat primul cuvânt din dicționar care conține, pe poziția respectivă, una din literele înscrise pe tasta acționată.

Cuvântul (dacă există în dicționar) va fi format luând literele de pe pozițiile corespunzătoare din fiecare cuvânt afișat, în ordine. Fiecare literă a unui cuvânt care nu este în dicționar va fi înlocuită cu un caracter *. Dacă există mai multe cuvinte care corespund unei anumite secvențe de acționări de taste, va fi afișat primul.

Se vor verifica toate posibilitățile de formare a unui cuvânt conform regulilor de mai sus. Aranjarea literelor pe tastatura telefonului mobil este următoarea:

1	2	3	4	5	6	7	8	9
Spațiu	ABC	DEF	GHI	JKL	MNO	PQRS	TUV	WXYZ

Sarcina voastră este de a realiza un program care să simuleze sistemul T9 pe baza unui dicționar dat.

Date de intrare

Fișierul de intrare **T9.IN** conține pe prima linie numărul M al cuvintelor din dicționar, pe următoarele M linii se află cuvintele din dicționar (unul pe fiecare linie). Următoarea linie conține numărul N al tastelor care s-au acționat, iar ultima linie conține cifrele înscrise pe cele N taste în ordinea apăsării, separate printr-un singur spațiu.

Date de ieșire

Fișierul de ieșire **T9.OUT** va conține pe o singură linie mesajul generat de sistemul T9 în urma acționării celor N taste.

Restricții și precizări

- $1 \le M, N \le 100$;
- cuvintele sunt ordonate alfabetic și sunt scrise cu majuscule;
- lungimea unui cuvânt nu depășește 100 de caractere.

Exemple

T9.IN
```
3
ABC
BBB
DEF
10
2 2 2 1 2 3 1 2 2 2
```

T9.OUT
```
ABC ** ABC
```

T9.IN
```
4
AZI
ESTE
IERI
VINERI
11
2 9 4 1 4 3 7 4 1 7 4
```

T9.OUT
```
AZI IERI **
```

Timp de execuție: 1 secundă/test

Clasa a X-a

P050237: Cadou

(prof. Mihaela Flutur)

De *Crăciun*, un grup de n copii s-au gândit să facă un cadou copiilor unei familii sărace care locuiește pe aceeași stradă cu ei. Fiecare copil a adus din casă cel puțin o portocală și cel mult zece portocale. Copiii din aceeași familie (pe care îi vom considera frați) aduc fiecare același număr de portocale. Cadoul constă din cele p portocale adunate.

Sarcina voastră este de a realiza un program care, cunoscând numărul de copii, relațiile dintre aceștia și numărul de portocale, determină câte portocale a adus fiecare copil. Se vor genera toate soluțiile, în ordine lexicografică.

Date de intrare

Fișierul **CADOU.IN** conține pe prima linie un număr întreg pozitiv n care reprezintă numărul de copii; pe fiecare dintre următoarele n linii se află un număr natural x_i care reprezintă numărul de frați ai copilului i și dacă este diferit de 0 este urmat de x_i numere care reprezintă frații copilului i. Datele de pe aceste n linii sunt despărțite printr-un singur spațiu. Ultima linie din fișier conține un număr p care reprezintă numărul total de portocale.

Date de ieșire

Fișierul de ieșire **CADOU.OUT** va conține câte o linie pentru fiecare soluție generată. O soluție este formată din n valori separate prin spații de forma: $nr_1\ nr_2 \ldots nr_n$, unde nr_i reprezintă numărul de portocale aduse de copilul i.

Restricții și precizări

- $1 \le n \le 30$;
- $1 \le p \le 50$;
- există cel mult zece familii.

Exemplu

CADOU.IN
```
3
1 2
1 1
0
5
```

CADOU.OUT
```
1 1 3
2 2 1
```

Timp de execuție: 1 secundă/test

P050238: Masă

(prof. Emanuela Cerchez)

O moleculă poate fi definită ca o secvență de atomi și poate fi reprezentată printr-o formulă chimică. De exemplu, litera **H** desemnează un atom de hidrogen, litera **C** desemnează un atom de carbon, **O** desemnează un atom de oxigen. Formula **COOH** reprezintă molecula formată dintr-un atom de carbon, doi atomi de oxigen și un atom de hidrogen.

Pentru a scrie formule în mod eficient, utilizăm regulile următoare.

Literele ce desemnează unii atomi pot fi grupate între paranteze. De exemplu formula **CH(OH)** conține gruparea **OH**. Grupările pot fi imbricate, adică o grupare poate conține alte grupări. Pentru a simplifica formula, aparițiile consecutive ale aceleiași litere pot fi înlocuite cu litera respectivă urmată de numărul de apariții. De exemplu, formula **COOHHH** poate fi scrisă **CO2H3** și reprezintă o moleculă constituită dintr-un atom de carbon, doi atomi de oxigen și 3 atomi de hidrogen.

De asemenea, aparițiile consecutive ale aceleiași grupări pot fi înlocuite de gruparea respectivă, urmată de numărul ei de apariții. De exemplu, formula **CH(CO2H)(CO2H)(CO2H)** poate fi scrisă **CH(CO2H)3**.

Molecula reprezentată de această formulă constă din patru atomi de carbon, patru atomi de hidrogen și șase atomi de oxigen.

Masa unei molecule este egală cu suma maselor atomilor din care este constituită.

Sarcina voastră este de a realiza un program care, pe baza unei formule chimice în care apar numai atomi de hidrogen, oxigen și carbon, calculează masa moleculei respective.

Date de intrare

Fișierul de intrare **MASA.IN** conține pe prima linie o formulă chimică construită după regulile de mai sus.

Date de ieșire

Fișierul de ieșire **MASA.OUT** va conține o singură linie pe care se va afla un număr care reprezintă masa moleculei din fișierul de intrare.

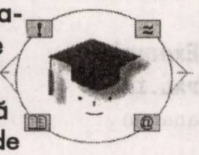
Restricții și precizări

- lungimea unei formule nu depășește 100 de caractere;
- masa unei molecule este un număr natural ce nu depășește 10000;
- un atom de hidrogen are masa 1, un atom de carbon are masa 12, iar un atom de oxigen are masa 16;
- numărul care indică repetițiile unei litere sau ale unei grupări este întotdeauna mai mare decât 1 și strict mai mic decât 10.

Exemple

MASA.IN	MASA.OUT
COOH	45

Despre concurs...



Concursul de programare "Urmașii lui Moisil" a fost organizat de Centrul de Excelență pentru Tineri Capabili de Performanță din Iași, în colaborare cu Liceul de Informatică "Grigore C. Moisil".

La concurs au participat 50 de elevi din șapte județe ale țării, reprezentanți ai centrelor de excelență din zona Moldovei. Centrul regional Iași a fost reprezentat de elevi din județele Bacău, Iași, Neamț și Suceava, în timp ce reprezentanții centrului regional Galați au fost elevi din județele Brăila, Galați și Vrancea.

Cele șase probleme cu care s-au confruntat concurenții au fost propuse de cadre didactice de la Liceul de Informatică din Iași.

Mai multe informații despre acest concurs puteți găsi pe pagina Web dedicată centrului de excelență Iași: http://www.liis.ro/~cex_is/. De asemenea, pe această pagină, veți putea găsi o mulțime de informații referitoare la activitatea centrului de excelență din Iași.

Redacția GInfo mulțumește d-nei prof. Emanuela Cerchez pentru informațiile furnizate despre acest concurs de programare.

MASA.IN	MASA.OUT
CH(CO2H)3	148

MASA.IN	MASA.OUT
((CH)2(OH2H)(C(H))O)3	222

Timp de execuție: 1 secundă/test

Clasele a XI-a și a XII-a

P050239: Palindrom

(clasele a XI-a și a XII-a)

Un palindrom este un cuvânt care se citește la fel și de la stânga la dreapta și de la dreapta la stânga. Dacă un cuvânt nu este palindrom, el poate fi tăiat în mai multe părți care să fie palindroame.

Sarcina voastră este de a realiza un program care să determine numărul minim de palindroame în care poate fi împărțită o secvență de caractere dată.

Date de intrare

Fișierul de intrare **PAL.IN** conține o singură linie pe care se află o secvență de caractere formată din litere mici.

Date de ieșire

Fișierul de ieșire **PAL.OUT** va conține o singură linie pe care se va afla numărul minim de palindroame.

Restricție
- numărul de caractere conținute de o secvență este de cel mult 100.

Exemple

PAL.IN	PAL.OUT
anaban	2

PAL.IN	PAL.OUT
abaccbcb	3

Explicații
Secvența anaban poate fi împărțită în două palindroame: a și naban. Secvența abaccbcb poate fi împărțită în trei palindroame: aba, cc și bcb.

Timp de execuție: 1 secundă/test

P050240: Loc

(prof. Emanuela Cerchez)

Elevii de la *Liceul de Informatică* vor să își aleagă un loc de întâlnire. Ei au luat harta orașului *Iași* și au marcat punctele culturale interesante (*Casa Pogor*, *Bolta Rece*, etc.), apoi le-au numerotat de la 1 la N, *Liceul de Informatică* având numărul 0. Între oricare două puncte culturale există cel mult o stradă, care poate fi parcursă în ambele sensuri.

Pentru a avea cât mai multe variante de a evita întâlniri nedorite, elevii au hotărât să aleagă drept loc de întâlnire un punct pentru care există un număr maxim de drumuri de lungime minimă care pleacă de la liceu.

Sarcina voastră este de a realiza un program care să aleagă un loc de întâlnire pentru elevii de la *Liceul de Informatică*.

Date de intrare
Fișierul de intrare **LOC.IN** are următoarea structură:
- prima linie conține un număr natural N care reprezintă numărul de puncte culturale interesante;
- a doua line conține numărul m al străzilor;
- fiecare dintre următoarele m linii conține câte trei numere naturale (x_i, y_i, L_i) separate printr-un singur spațiu cu semnificația: *există o stradă de lungime L_i între punctele x_i și y_i.*

Date de intrare
Fișierul de ieșire **LOC.OUT** va conține trei linii. Pe prima dintre ele se va afla numărul p al locului ales pentru întâlnire. A doua linie va conține numărul de drumuri de lungime minimă de la *Liceul de Informatică* până la locul de întâlnire, iar pe cea de-a treia linie se va afla un număr care reprezintă distanța minimă de la liceu până la locul de întâlnire.

Restricții și precizări
- $N \leq 100$;
- $x_i, y_i \in \{0, 1, ..., N\}, \forall\, i \in \{1, 2, ..., m\}$;
- $L_i \in \mathbb{N}^*, 0 < L_i < 10000, \forall\, i \in \{1, 2, ..., m\}$;
- dacă există mai multe soluții, trebuie determinată doar una dintre ele.

Exemplu

LOC.IN	LOC.OUT
4	4
6	3
0 1 10	40
0 4 40	
1 2 10	
2 3 10	
1 4 30	
2 4 20	

Timp de execuție: 1 secundă/test

Premianții

Clasa a IX-a
Daniel Ghiță, Suceava - premiul I
Alexandru Diaconu, Iași - premiul II
Iolanda Popa, Iași - premiul III
Andrei Aiordăchioaiei, Galați - mențiune
Ionuț Fechete, Suceava - mențiune

Clasa a X-a
Horia-Andrei Chiochină, Iași - premiul I
Andrei Blaj, Suceava - premiul I
Emilian Miron, Bacău - premiul III
Eugen Marcu, Galați - premiul III
Constantin Dumitrașcu, Vrancea - premiul III
Cristian Bîrlădeanu, Iași - mențiune
Adrian Apostol, Iași - mențiune
Vlad Olaru, Neamț - mențiune

Clasele a XI-a și a XII-a
Ștefăniță Fechete, Suceava - premiul I
Andrei Benea, Vrancea - premiul I
Adrian Airinei, Bacău - premiul II
Mihai Chiru, Brăila - premiul III
Codrin Grăjdeanu, Galați - mențiune
Tudor Mihordea, Iași - mențiune
Lucian-Daniel Stanciu, Brăila - mențiune
Alexandru Dabija, Iași - mențiune
Ștefan Ciobâcă, Suceava - mențiune

Centrul de excelență Iași

Bursele AGORA

Prima fază a concursului de programare Bursele Agora s-a încheiat. În cadrul acestui număr vă vom prezenta soluțiile problemelor de la ultima rundă, clasamentul etapei, precum și clasamentul final.

A cincea rundă

Aceasta a fost cea mai puțin dificilă rundă a concursului nostru; 18 participanți au reușit să obțină toate cele 300 de puncte puse în joc. Lista primilor clasați este:

1. Csaba András, Oradea	*300*
1. Liviu-Cosmin Andreicuț, Bacău	*300*
1. Adrian Balașko, Zalău	*300*
1. Bogdan Beldie, Oradea	*300*
1. Andrei Benea, Focșani	*300*
1. Radu Berinde, București	*300*
1. Cristian Bojinovici, București	*300*
1. Mihai-Daniel Cilidariu, Botoșani	*300*
1. Ștefan Ciobâcă, Suceava	*300*
1. Victor-Marius Costan, București	*300*
1. Tiberiu Dăneț, București	*300*
1. Radu Dondera, București	*300*
1. Ștefăniță Fechete, Suceava	*300*
1. Andrei Giurgiu, București	*300*
1. Maximilian Machedon, București	*300*
1. Silvestru-Cosmin Negrușeri, Bistrița	*300*
1. Constantin Răuțu, Botoșani	*300*
1. Alexandru-Emilian Șușu, București	*300*
19. Claudiu Raicu, Pitești	*296*
20. Victoria Tarasova, Ucraina	*295*
21. Csaba Pătcaș, Oradea	*290*
22. Bogdan Nicolae, Sibiu	*289*
23. Mihail-Cosmin Pițt-Rada, Dr. Tr. Severin	*288*
24. Paul Rusu, Cluj-Napoca	*287*
25. Mugurel-Ionuț Andreica, București	*285*
26. Marius-Dorin Moraru, Buftea	*277*
27. Radu-Andrei Ștefan, Brașov	*269*
28. Ștefan Gheorghe, București	*264*
28. Andrei-Eugeniu Ioniță, Slatina	*264*
30. Ștefan Bucur, București	*258*
31. Ioan-Claudiu Gruia, București	*243*
31. Andrei Vancea, Cluj-Napoca	*243*
33. Alexandru Ivan, București	*213*
34. Andrei David, Râmnicu Vâlcea	*209*
35. Valentin Bran, Satu Mare	*200*
35. Dorin-Daniel Dudău, Târgu-Jiu	*200*
35. Octavian-Daniel Dumitran, București	*200*
35. Valentin Gușă, București	*200*
35. Andrei Homescu, Târgu-Jiu	*200*
35. Flaviu Pașca, Bistrița	*200*

Etapa finală

Runda a șasea va avea loc la *Liceul "Emil Racoviță"* din *Cluj-Napoca* pe data de 1 iunie 2002. La această etapă vor participa primii zece clasați în clasamentul general după primele cinci etape.

Problemele rundei finale vor fi disponibile pe *site*-ul revistei noastre în ziua concursului, începând cu ora 10:00. Participanții care nu s-au calificat la etapa finală, dar doresc să cunoască punctajele pe care le-ar fi obținut, pot trimite rezolvările până la ora 15:00. Modalitatea de trimitere a soluțiilor și modul de construire a arhivei sunt aceleași cu cele de la primele cinci runde.

Runda finală va fi considerată ca fiind a șasea etapă a concursului, deci arhiva pe care o veți trimite se va numi `XXXXR6.ZIP` sau `XXXXR6.RAR`, unde `XXXX` este codul dumneavoastră de identificare.

La mulți ani!

Cristian Alexandrescu, Botoșani - 3 V 1982
Tudor Salomie, Cluj-Napoca - 4 V 1984
Tiberiu Dăneț, București - 4 V 1985
Radu-Andrei Ștefan, Brașov - 6 V 1981
Adrian Cearnău, București - 6 V 1985
Alexandru Palade, Târgu Mureș - 7 V 1984
Ștefan Gheorghe, București - 10 V 1985
Andreea Rădulescu, Bragadiru - 13 V 1982
Lorena-Iulia Dăian, Cluj-Napoca - 13 V 1985
Bogdan Marincaș, Cluj-Napoca - 13 V 1985
Claudiu Raicu, Pitești - 15 V 1985
Constantin Răuțu, Botoșani - 16 V 1984
Alexandru-Ioan Tomescu, București - 18 V 1985
Florin Buga, Slatina - 20 V 1983
Bogdan Giușcă, București - 20 V 1983
Marius-Constantin Muja, Orăștie - 21 V 1981
Dan Creivean, Alba Iulia - 25 V 1984
Cristian Bojinovici, București - 28 V 1980
Ștefan Ciobâcă, Suceava - 28 V 1984
Dan-Octavian Savu, București - 29 V 1983
George-Andrei Dunca, Cluj-Napoca - 29 V 1984

Runda 5, problema 1

FIBONACCI

Această problemă a fost rezolvată corect de 41 dintre participanții la a cincea rundă a concursului *Bursele Agora*. Ea nu ridica dificultăți deosebite; putea fi rezolvată folosind o simplă formulă matematică.

Se observă că $F_6 = 8$ se divide cu 2^3, $F_{12} = 144$ se divide cu 2^4 și $F_{24} = 46368$ se divide cu 2^5. De aici rezultă o regulă destul de clară: indicele elementului șirului lui *Fibonacci* divizibil cu 2^3 este $6 = 3 \cdot 2$, cel al elementului divizibil cu 2^4 este $12 = 3 \cdot 2^2$, iar cel al elementului divizibil cu 2^5 este $24 = 3 \cdot 2^3$. Putem presupune că elementul șirului lui *Fibonacci* divizibil cu 2^n are indicele $3 \cdot 2^{n-2}$.

În cele ce urmează vom demonstra că această afirmație este adevărată. Pentru aceasta vom folosi următoarea proprietate a șirului lui *Fibonacci*: elementul $F_{k \cdot n}$ este divizibil cu elementul F_n. Această proprietate poate fi demonstrată foarte ușor, folosind expresia generală a termenilor șirului:

$$F_n = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}.$$

Din motive de spațiu, nu vom prezenta aici toate calculele care trebuie efectuate pentru a demonstra proprietatea.

Utilizând acest rezultat matematic, vom aplica metoda inducției matematice pentru a demonstra că elementul șirului lui *Fibonacci* divizibil cu 2^n are indicele $3 \cdot 2^{n-2}$.

Ipoteza de inducție va fi: *elementul șirului lui Fibonacci divizibil cu* 2^{n-1} *are indicele* $3 \cdot 2^{n-3}$.

Pentru $n = 4$ ipoteza de inducție poate fi verificată foarte ușor, deoarece elementul cu indicele $3 \cdot 2^{4-3} = 6$ este 8 și acesta este divizibil cu $2^{4-1} = 8$.

Folosind ipoteza de inducție vom demonstra afirmația inițială. Datorită faptului că $F_{2 \cdot n}$ este divizibil cu elementul F_n, elementul cu indicele $3 \cdot 2^{n-2}$ va fi divizibil cu 2^{n-1}.

Mai mult, raportul dintre elementul cu indicele $3 \cdot 2^n$ și cel cu indicele $3 \cdot 2^{n-1}$ este întotdeauna un număr par (proprietatea se poate verifica tot pe baza expresiei generale a termenilor șirului), deci elementul cu indicele $3 \cdot 2^{n-2}$ va fi divizibil cu $2 \cdot 2^{n-1} = 2^n$.

Așadar, am demonstrat că există un element al șirului lui *Fibonacci* divizibil cu 2^n care are indicele $3 \cdot 2^{n-2}$. Mai trebuie arătat că acest indice este mai mic decât 2^n. Este evident că $3 \cdot 2^{n-2} < 4 \cdot 2^{n-2} = 2^n$ pentru $n > 2$. Așadar, pentru $n > 2$, soluția problemei este dată de expresia $3 \cdot 2^{n-2}$.

Codul sursă al programului este disponibil pentru *download* la adresa **www.ginfo.ro/concurs/fibonacci.shtml**.

Au rezolvat corect:

Csaba András, Oradea
Mugurel-Ionuț Andreica, București
Liviu-Cosmin Andreicuț, Bacău
Adrian Balașko, Zalău
Cristian Băicoianu, Ploiești
Bogdan Beldie, Oradea
Andrei Benea, Focșani
Radu Berinde, București
Cristian Bojinovici, București
Valentin Bran, Satu Mare
Ștefan Bucur, București
Mihai-Daniel Cilidariu, Botoșani
Ștefan Ciobâcă, Suceava
Victor-Marius Costan, București
Tiberiu Dănet, București
Radu Dondera, București
Dorin-Daniel Dudău, Târgu-Jiu
Octavian-Daniel Dumitran, București
Ștefăniță Fechete, Suceava
Ștefan Gheorghe, București
Andrei Giurgiu, București
Valentin Gușă, București
Bogdan-Vasile Hârjoc, Cugir
Andrei Homescu, Târgu-Jiu
Andrei-Eugeniu Ioniță, Slatina
Alexandru Ivan, București
Maximilian Machedon, București
Emilian Miron, Bacău
Marius-Dorin Moraru, Buftea
Silvestru-Cosmin Negruțeri, Bistrița
Bogdan Nicolae, Sibiu
Flaviu Pașca, Bistrița
Csaba Pătcaș, Oradea
Alpár-Ferenc Perini, Sfântu Gheorghe
Mihail-Cosmin Pițu-Rada, Dr. Tr. Severin
Claudiu Raicu, Pitești
Constantin Răuțu, Botoșani
Paul Rusu, Cluj-Napoca
Radu-Andrei Ștefan, Brașov
Alexandru-Emilian Șușu, București
Victoria Tarasova, Ucraina

Runda 5, problema 2

ATLANTIS

Această problemă a fost rezolvată corect de 39 dintre concurenții care au participat la runda a cincea a concursului *Bursele Agora*. Soluția era foarte ușor de găsit, fiind suficientă o simplă observație matematică.

Pentru început, vom arăta că există o secvență de patru semne a cărei prezență va duce la creșterea rezultatului cu 4. Vom presupune că există o secvență de m semne care duce la obținerea rezultatului R. Dacă adăugăm cele patru semne, vom obține o secvență de $m + 4$ semne care va duce la obținerea rezultatului $R + 4$. Secvența respectivă este:

+−−+

Se observă că, prin adăugarea acestei secvențe, valoarea rezultatului va crește cu $m^2 - (m + 1)^2 - (m + 2)^2 + (m + 3)^2$. Efectuând calculele obținem:

$m^2 - (m^2 + 2 \cdot m + 1) - (m^2 + 4 \cdot m + 4) + (m^2 + 6 \cdot m + 9) =$
$m^2 - m^2 - 2 \cdot m - 1 - m^2 - 4 \cdot m - 4 + m^2 + 6 \cdot m + 9 = 4.$

Așadar, prin adăugarea secvenței +−−+ la sfârșitul unei secvențe date, valoarea rezultatului crește cu 4.

Se observă că, dacă dorim să obținem o valoare care este multiplu de 4 (are forma $4 \cdot k$), va trebui să folosim k secvențe de această formă.

În cele ce urmează vom arăta modul în care pot fi obținute secvențe de semne care să ducă la rezultate de forma $4 \cdot k + 1$, $4 \cdot k + 2$ și $4 \cdot k + 3$.

Se observă că, pentru a obține secvențe care au forma $4 \cdot k + p$, va trebui să adăugăm k secvențe +−−+ la secvențele care duc la obținerea rezultatului p.

Ca urmare, va trebui să căutăm secvențe de semne care să ducă la obținerea rezultatelor 1, 2 și 3.

Pentru a obține rezultatul 1, secvența constă dintr-un singur semn +. O secvență formată doar din acest semn duce la obținerea rezultatului $0 + 1^2 = 1$. Așadar, dacă se caută rezultate de forma $4 \cdot k + 1$, secvențele de semne au următoarea structură: +−−+...+−−+.

Pentru a obține rezultatul 2, secvența constă din patru semne: trei semne −, urmate de un semn +. O secvență formată doar din aceste semne duce la obținerea rezultatului $0 - 1^2 - 2^2 - 3^2 + 4^2 = 0 - 1 - 4 - 9 + 16 = 2$. Așadar, dacă se caută rezultate de forma $4 \cdot k + 2$, secvențele de semne au următoarea structură: −−−+−−+...+−−+.

Pentru a obține rezultatul 3, secvența constă din două semne: un semn −, urmat de un semn +. O secvență formată doar din aceste semne duce la obținerea rezultatului $0 - 1^2 + = 0 - 1 + 4 = 3$. Așadar, dacă se caută rezultate de forma $4 \cdot k + 3$, secvențele de semne au următoarea structură: −+−−+...+−−+.

Au rezolvat corect:

Csaba András, Oradea
Mugurel-Ionuț Andreica, București
Liviu-Cosmin Andreicuț, Bacău
Adrian Balaşko, Zalău
Bogdan Beldie, Oradea
Andrei Benea, Focșani
Radu Berinde, București
Cristian Bojinovici, București
Valentin Bran, Satu Mare
Mihai-Daniel Cilidariu, Botoșani
Ștefan Ciobâcă, Suceava
Victor-Marius Costan, București
Andrei David, Râmnicu Vâlcea
Tiberiu Dănet, București
Radu Dondera, București
Dorin-Daniel Dudău, Târgu-Jiu
Octavian-Daniel Dumitran, București
Ștefăniță Fechete, Suceava
Ștefan Gheorghe, București
Andrei Giurgiu, București
Ioan-Claudiu Gruia, București
Valentin Guşă, București
Andrei Homescu, Târgu-Jiu
Andrei-Eugeniu Ioniță, Slatina
Maximilian Machedon, București
Marius-Dorin Moraru, Buftea
Silvestru-Cosmin Negruşeri, Bistrița
Bogdan Nicolae, Sibiu
Flaviu Pașca, Bistrița
Csaba Pătcaș, Oradea
Mihail-Cosmin Pițigoi-Rada, Dr. Tr. Severin
Claudiu Raicu, Pitești
Constantin Răuțu, Botoșani
Flaviu Roman, Cluj-Napoca
Paul Rusu, Cluj-Napoca
Radu-Andrei Ștefan, Brașov
Alexandru-Emilian Șușu, București
Victoria Tarasova, Ucraina
Andrei Vancea, Cluj-Napoca

Codul sursă al programului care implementează algoritmul descris este disponibil pentru *download* la adresa **www.ginfo.ro/concurs/atlantis2.shtml**.

Runda 5, problema 3

TIC-TAC-TOE

După părerea unui număr destul de mare de concurenți, aceasta a fost cea mai interesantă problemă propusă spre rezolvare la această ediție a concursului de programare *Bursele Agora*.

Având în vedere restricția de timp impusă, nu era foarte dificilă scrierea unor programe care să joace 100.000 de partide într-un interval de o secundă și să obțină remize la fiecare dintre acestea. Pentru această problemă 23 de concurenți au reușit să obțină 100 de puncte. Clasamentul celor mai rapide programe (în funcție de numărul de remize obținute) este prezentat în caseta alăturată.

Programul fiecărui concurent a fost rulat de cinci ori și a fost luat în considerare cel mai bun rezultat obținut.

Modulul care reprezenta primul jucător folosea o strategie care să îi permită să nu piardă nici o partidă. Acesta nu încerca să câștige jocul chiar dacă avea posibilitatea.

Poziția în care primul jucător amplasează un `'X'` este determinată pe baza unei funcții. Pentru fiecare mutare posibilă este evaluată valoarea funcției și este aleasă mutarea pentru care această valoare este maximă. Funcția folosită este:

$(x_4 \cdot x_1 + x_2) \cdot x_7 - (x_1 - x_2 + x_7 \cdot x_5) - (x_8 - (x_3 \cdot x_5))$,

unde x_i indică valoarea corespunzătoare stării celei de-a i-a căsuțe a tablei. Căsuțele sunt numerotate astfel:

0	1	2
3	4	5
6	7	8

În cazul în care căsuța i este ocupată cu un `'X'`, valoarea x_i va fi 5. Dacă în căsuță se află un `'0'`, valoarea x_i va fi -5. În situația în care căsuța este liberă, valoarea x_i va fi 2.

Expresia de mai sus, precum și valorile corespunzătoare stărilor căsuțelor, au fost obținute cu ajutorul unui algoritm evolutiv.

Există diverse metode de rezolvare pentru această problemă. Cea mai rapidă dintre ele se bazează pe faptul că numărul configurațiilor posibile este relativ mic. Dacă luăm în considerare toate configurațiile în care se poate afla, observăm că numărul acestora este $3^9 = 19683$. Practic, numărul configurațiilor valide este mai mic, deoarece o parte dintre cele 19683 de configurații nu pot fi obținute dacă se respectă regulile jocului.

Așadar, pot fi determinate în prealabil cele mai bune mutări care trebuie efectuate în toate situațiile posibile. Acestea vor fi păstrate cu ajutorul unor tablouri de constante.

Pentru a nu efectua multe operații în momentul realizării unei mutări, indicele configurației următoare nu va fi determinat pe baza valorilor corespunzătoare căsuțelor tablei, ci pe baza indicelui configurației curente și a mutării efectuate.

Aplicând această strategie era relativ ușor de obținut un program care să poată juca cel puțin 150.000 de partide.

Există multe alte soluții care ar fi obținut 100 de puncte. Cea mai interesantă dintre ele este folosirea unei expresii asemănătoare cu cea folosită de modulul care reprezintă primul jucător.

Codul sursă al unui program care obține 100 de puncte este disponibil pentru *download* la adresa **www.ginfo.ro/concurs/tictactoe.shtml**.

Rezultate Tic-Tac-Toe:

1.	Maximilian Machedon, București	162839
2.	Victor Costan, București	160242
3.	Andrei Benea, Focșani	158414
4.	Radu Berinde, București	157082
5.	Alexandru Șușu, București	145658
6.	Ștefăniță Fechete, Suceava	141908
7.	Ștefan Ciobâcă, Suceava	139146
8.	Andrei Giurgiu, București	134671
9.	Bogdan Beldie, Oradea	125336
10.	Csaba András, Oradea	122884
11.	Radu Dondera, București	117201
12.	Liviu-Cosmin Andreicuț, Bacău	116434
13.	Ioan-Claudiu Gruia, București	116251
14.	Alexandru Ivan, București	109737
15.	Adrian Balașko, Zalău	109031
16.	Mihai Cilidariu, Botoșani	108850
17.	Cristian Bojinovici, București	107654
18.	Cosmin Negrușeri, Bistrița	106429
19.	Tiberiu Dănet, București	106274
20.	Constantin Răuțu, Botoșani	105756
21.	Andrei David, Râmnicu Vâlcea	104414
22.	Andrei Vancea, Cluj-Napoca	102808
23.	Gheorghe Fan, Ștefești	100835

GInfo Top 50

1. Mugurel-Ionuț Andreica, București	1424
2. Silvestru-Cosmin Negrușeri, Bistrița	1390
3. Tiberiu Dăneț, București	1218
4. Maximilian Machedon, București	1149
5. Radu Dondera, București	1122
6. Adrian Balașko, Zalău	1108
7. Radu Berinde, București	1106
8. Csaba András, Oradea	1104
9. Claudiu Raicu, Pitești	1103
10. Octavian-Daniel Dumitran, București	1102
11. Paul Rusu, Cluj-Napoca	1091
12. Bogdan Nicolae, Sibiu	1058
13. Csaba Pătcaș, Oradea	1042
14. Radu-Andrei Ștefan, Brașov	1040
15. Ștefăniță Fechete, Suceava	999
16. Andrei Vancea, Cluj-Napoca	982
17. Liviu-Cosmin Andreicuț, Bacău	952
18. Victor-Marius Costan, București	884
19. Andrei Homescu, Târgu-Jiu	870
20. Ioan-Claudiu Gruia, București	846
21. Andrei Giurgiu, București	763
22. Adrian-Nicolae Cârcu, Bistrița	714
23. Ștefan Gheorghe, București	680
24. Victoria Tarasova, Ucraina	678
25. Marius-Dorin Moraru, Buftea	665
25. Constantin Răuțu, Botoșani	665
27. Mihai-Daniel Cilidariu, Botoșani	644
28. Ștefan Bucur, București	625
28. Emilian Miron, Bacău	625
30. Andrei Benea, Focșani	613
31. Mihail-Cosmin Pițˌ-Rada, Dr. Tr. Severin	609
32. Andrei David, Rm. Vâlcea	580
33. Cristian Bojinovici, București	579
34. Flaviu Pașca, Bistrița	566
35. Alpár-Ferenc Perini, Sfântu Gheorghe	550
36. Cristian Băicoianu, Ploiești	532
37. Bogdan-Vasile Hârjoc, Cugir	518
38. Bogdan Beldie, Oradea	479
39. Valentin Gușă, București	469
40. Alexandru Ivan București	422
41. Andrei-Eugeniu Ioniță, Slatina	414
42. Flaviu Roman, Cluj-Napoca	383
43. Slotta Johannes, Germania	375
43. Aurelian Marinescu, Brașov	375
43. Alexandru-Emilian Șușu, București	375
46. Dorin-Daniel Dudău, Târgu-Jiu	359
47. Liviu-Laurențiu Iacob, București	332
48. Sorin Otescu, Brașov	310
49. Adrian Airinei, Bacău	306
50. Ștefan Ciobâcă, Suceava	300

Tipuri de REȚELE

Mihai Scorțaru

În cadrul acestui articol vom prezenta câteva dintre caracteristicile anumitor tipuri de rețele locale. Vor fi descrise câteva dintre cele mai cunoscute standarde și anume: 10Base2, 10Base5, 10Broad36, FOIRL, 10BaseT, 10BaseFP, 10BaseFB și 10BaseFL.

Standardul 10Base2

Acest standard prezintă caracteristicile *transceiver*-elor și mediilor de transmisie care trebuie utilizate pentru a se obține o viteză de transmisie de 10 Mbps, folosindu-se segmente de cablu coaxial subțire de 185 m.

Caracteristicile principale ale cablului folosit sunt:

- diametrul de 0,2 inci;
- impedanța cuprinsă între 48 și 52 Ω;
- viteza minimă de propagare a semnalului electric de 0,65 · *c* (*c* reprezintă viteza luminii).

Acest cablu este foarte ieftin și poate fi instalat foarte ușor. Din aceste motive, o astfel de rețea este cunoscută și sub denumirea ***Cheapernet***. Câteva dintre regulile de configurare specificate de standard sunt:

- atenuarea maximă pe segment este de cel mult 8,5 dB pentru o frecvență de 10 MHz și de cel mult 6 dB pentru o frecvență de 5 MHz;
- lungimea maximă a unui segment este de 185 m;
- distanța dintre două *transceiver*-e trebuie să fie de cel puțin 0,5 m;
- la un segment pot fi legate cel mult 30 de stații;
- pot exista cel mult cinci segmente;
- pentru blocarea interferențelor este obligatorie folosirea unor terminatori de segment cu impedanța de 50 Ω.

Standardul 10Base5

Acest standard prezintă caracteristicile unei rețele cu viteza de 10 Mbps, construită folosind segmente de cablu cu lungimea de cel mult 500 m.

Acest standard specifică, practic, aceleași elemente ca și pentru rețelele *Ethernet*. Caracteristicile *transceiver*-elor, ale conectorilor și ale mediilor de transmisie sunt identice cu cele specificate de standardul *Ethernet v*2.0.

De asemenea, nu există diferențe între regulile de configurare ale rețelelor 10*Base*5 și *Ethernet*.

Standardul 10Broad36

În accepțiunea actuală, rețelele cu transmisie în bandă largă (*broadband*) sunt echivalente cu rețelele cu transmisie analogică. Cablul folosit este identic cu cel utilizat pentru televiziunea prin cablu.

Așadar, se folosește un cablu coaxial cu impedanța de 75 Ω care permite transmisii cu frecvențe de până la 450 MHz și lungimi mai mari ale segmentelor. Aceste performanțe superioare pot fi atinse datorită utilizării semnalizării analogice, care este mai puțin sensibilă decât semnalizarea digitală.

Unul dintre cele mai importante inconveniente este dat de imposibilitatea de a transfera pachete în ambele sensuri folosind un singur cablu de legătură. Aceasta se datorează faptului că amplificatoarele analogice folosite pentru acest tip de rețea transmit semnalele într-o singură direcție. Pentru rezolvarea acestei probleme ar trebui folosite două cabluri pentru realizarea transmisiei în ambele sensuri. O altă soluție o constituie folosirea a două benzi distincte de frecvențe pentru emisie și recepție.

Rețelele cu transmisie în bandă largă au o arhitectură arborescentă. Rădăcina arborelui poartă denumirea de ***terminator***, în timp ce nodurile sunt numite ***divizoare***.

Rolul nodurilor este de a transmite semnalul primit de la predecesor către toți descendenții direcți. Pentru realizarea interfeței cu mediul fizic sunt utilizate *transceiver*-e de bandă largă.

Pentru realizarea comunicării dintre două stații, semnalul trimis de emițător parcurge drumul spre rădăcină folosind linia de transmisie, și apoi este trimis de la rădăcină spre receptor folosindu-se linia de recepție. Lățimea de bandă a unui canal de transmisie este de 14 MHz, iar lungimea maximă a unui segment este de 3,6 km.

La nivelul accesului la mediu, apare o diferență față de rețelele de tip *Ethernet*. Cadrele conțin un câmp suplimentar numit *EOF* (*End Of Frame*) cu lungimea de 23 de biți și indică terminarea cadrului.

Datorită faptului că nu există posibilități de îmbunătățire semnificativă a performanțelor acestui tip de rețele, importanța rețelelor analogice a scăzut mult în ultima perioadă de timp.

Standardul FOIRL

Pentru a îmbunătăți performanțele rețelelor 10*Base*5 poate fi folosit un mediu de conectare bazat pe fibră optică. Acesta permite creșterea distanței maxime dintre două stații datorită vitezei mai mari de transmitere a semnalului. Noul standard poartă denumirea *FOIRL* (*Fiber Optic Inter Repeater Link*) și prezintă caracteristicile mediului de transmisie și a unității de atașare la mediu.

Unitatea de atașare la mediu este numită *FOMAU* (*Fiber Optic Medium Access Unit*) și este capabilă să transmită și să recepționeze semnale optice pe segmente cu lungimea de cel mult 1000 m. Ea este formată dintr-o interfață independentă de mediu (*FOPMA - Fiber Optic Physical Medium Attachment*) și una dependentă de mediu (*FOMDI - Fiber Optic Medium Dependent Interface*).

Componenta dependentă de mediu conține elementele de transmisie și recepție pentru fibra optică; lungimea de undă folosită este de 850 nm, iar fibra optică și componentele pasive trebuie să respecte standardul EIA/TIA 568.

Componenta independentă de mediu are aceleași funcții ca și componenta echivalentă de la rețelele *Ethernet*, la care se adaugă funcțiile specifice noului mediu de transmisie:

- transmiterea fluxului de biți de la repetorul 10*Base*5 către cablul de fibră optică;
- recepția fluxului de biți de la fibra optică și transmiterea sa către repetor;
- determinarea prezenței semnalului de coliziune și interpretarea corectă a acestuia;
- determinarea prezenței erorilor de tipul transmisiilor prea lungi sau a lungimilor necorespunzătoare ale cadrelor și întreruperea transmisiei în cazul apariției acestora;
- detectarea nivelului critic de semnal pe fibră și întreruperea corespunzătoare a recepției.

Mediul de transmisie este cablul de fibră optică; este folosită fibra multimodală cu dimensiuni de 62,5/125.

Standardul 10BaseT

Acest standard constituie prima abordare structurată a problemelor legate de dezvoltarea rețelelor; toate stațiile sunt legate la un *repetor multiport* (**hub**), nemaifiind atașate în locuri diferite.

Viteza de transmisie este de 10 Mbps și pentru transmiterea datelor este folosit cablul torsadat (*twisted pair*). Singura modalitate de conectare a două stații este legătura punct la punct.

Din acest motiv este necesară utilizarea repetoarelor multiport; astfel, apare o topologie stelară. La nivelul controlului accesului la mediu este implementat protocolul 802.3.

Unitatea de atașare la mediu are posibilitatea de a gestiona semnale electrice de-a lungul unor cabluri cu lungimea de cel mult 100 m.

Principalele caracteristici ale unui *transceiver* 10*BaseT* sunt următoarele:

- transmiterea datelor către mediu folosindu-se codificarea *Manchester*;
- recepția datelor provenite din mediu;
- detectarea semnalului de coliziune în mediu și trimiterea către interfață a semnalului corespunzător;
- generarea semnalelor de test pentru circuitele de detectare a coliziunii;
- detectarea cadrelor care au lungimea incorectă;
- transmiterea spre interfață a datelor care sunt trimise spre mediu;
- testarea integrității legăturilor.

Segmentele 10*BaseT* sunt formate din cablu torsadat cu cel puțin două perechi de fire răsucite. Principalele caracteristici ale cablului sunt:

- impedanța cuprinsă între 85 și 115 Ω;
- frecvența de până la 16 MHz;
- lungimea cuprinsă între 0,6 și 100 m;
- viteza de propagare a semnalului de cel puțin 0,585 · *c*;
- atenuarea de cel mult 11,5 dB.

Standardul 10BaseFP

Potrivit acestui standard (bazat pe fibră optică), pentru construirea rețelelor este folosită o topologie de tip stea pasivă. Aceasta implică existența unui element central și a unui ***splitter*** care divide și redirecționează semnalul.

Cea mai importantă caracteristică specificată de standard este distanța maximă de 1000 m între două unități de atașare la mediu. Această distanță este dată de suma lungimilor traseelor de la cele două stații la steaua pasivă.

Standardul 10BaseFB

O altă posibilitate de folosire a fibrei optice pentru construirea unei rețele locale este descrisă de standardul 10*BaseFB*. Acesta descrie condițiile care trebuie satisfăcute pentru ca fibra optică să poată fi folosită ca *backbone* (coloană vertebrală) între două repetoare construite pe baza standardului 802.3.

Transmiterea datelor este sincronă, ceea ce permite folosirea unor *transceiver*-e tolerante la erori. Aceste echipamente conțin două interfețe pentru accesul la mediu: una *principală* și una de *restaurare*, care intră în funcțiune dacă se detectează defectarea primeia.

Standardul 10BaseFL

Acest standard descrie modul în care poate fi folosită fibra optică pentru realizarea legăturii dintre repetoare sau stații ale unei rețele locale construite pe baza standardului 802.3. Legăturile pot fi de tip punct la punct sau stelare (dacă se folosesc repetoare multiport).

Bibliografie

1. Andrew S. Tannenbaun, *Rețele de calculatoare, Editura* Computer Press Agora, 1999
2. Vasile-Teodor Dădârlat, *Rețele de calculatoare,* Editura Albastră, 1999

Mihai Scorțaru este redactor-șef adjunct al GInfo. Poate fi contactat prin e-mail la adresa **skortzy@yahoo.com**.

INSTRUMENTE de programare NESECVENȚIALĂ

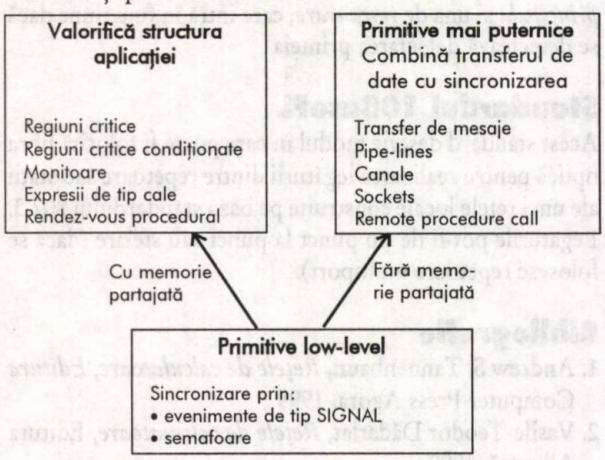
Anca Vasilescu, Oana Georgescu

În acest ultim episod al serialului despre problematica sistemelor nesecvențiale vom lua în discuție mecanismele specifice puse la dispoziție de limbajele de programare pentru implementarea aspectelor paralele, concurente sau distribuite evidențiate la nivelul diferitelor aplicații. Aceste instrumente de programare sunt cunoscute sub denumirea de mecanisme IPC, adică mecanisme de comunicare între procese (InterProcess Communication).

Tratarea mecanismelor *IPC* (de comunicare între procese) valorifică toate aspectele problematicii sistemelor nesecvențiale introduse în episoadele anterioare: probleme clasice specifice, clasificări, caracteristici.

Pentru programarea sistemelor paralele, concurente sau distribuite s-au dezvoltat o serie de mecanisme de implementare a concurenței la nivel *low-level* sau la nivelul limbajelor de programare de nivel înalt. Elementele discutate în acest context se vor axa pe prezentarea mecanismelor de comunicare între procese (*IPC - InterProcess Communication*), la diferite niveluri de abordare.

În figura următoare este reprezentată evoluția mecanismelor și primitivelor *IPC*.



Variantele *IPC* cu sau fără memorie partajată se regăsesc în două modalități de abordare a comunicării și anume: procesul activ poate să apeleze o procedură de interfață a unui modul pasiv (monitor), poate să trimită un mesaj în care formulează o cerere specifică. Aceste alternative se încadrează într-o clasificare mai generală a mecanismelor *IPC* de nivel înalt:

- mecanisme *IPC* pentru transmiterea de mesaje (*message passing*)
  - transmitere *asincronă* sau cu zone tampon;
  - transmitere *sincronă* sau fără zone tampon;
- mecanisme *IPC* procedurale
  - apel de procedură pentru obiecte pasive (se folosesc monitoare);
  - apel de procedură pentru obiecte active (mecanismul *rendez-vous*).

O clasă superioară de mecanisme *IPC* este dată de modalitățile de comunicare distribuită între procese (*Distributed IPC*). Prezența lor este necesară în sistemele *hard* și/sau *soft* distribuite și, implicit, trebuie privite și analizate prin prisma structurilor sistemelor pe care le deservesc. Din acest punct de vedere trebuie remarcat modelul *client-server* de implementare a aplicațiilor distribuite, care este în strânsă legătură atât cu suportul fizic și logic de comunicare disponibil în sistem, cât și cu limbajul pe care sistemul respectiv îl suportă. O alternativă la abordarea *client-server* este modelarea obiectuală a sistemelor. Aceasta aplică tehnologia programării orientate obiect la nivelul sistemelor prin interpretarea fiecărei resurse partajate ca un obiect [1].

Dintre mecanismele enumerate în clasificarea anterioară le vom lua în discuție pe cele mai bine reprezentate în limbajele de programare moderne și anume: *semafoarele*, *monitoarele*, *comunicarea prin transfer de mesaje* și *invo-*

carea la distanță (ca o generalizare a mecanismului de apel de procedură la distanță, *RPC*).

Semafoare

Noțiunea de *semafor* a fost introdusă în 1968 de ***Dijkstra*** împreună cu descrierea proiectului de sistem de operare numit *THE* (*Technische Hogeschool Eindhoven*). Sistemul *THE* proiectează o ierarhie de resurse virtuale gestionată de un sistem de procese strict organizate pe niveluri. [5]

Așa cum a fost introdus de *Dijkstra*, un ***semafor*** este un tip de date. Asupra unei variabile de tip semafor se pot executa operațiile `WAIT(SEMAFOR)` și `SIGNAL(SEMAFOR)` numite în acest context și ***primitive***, deoarece sunt definite ca operații atomice. În plus, asupra semafoarelor este permisă efectuarea unei singure operații, și anume inițializarea valorii variabilei semafor. Sintaxa acestor operații depinde de limbajul de programare în care se lucrează cu semafoare.

Reprezentarea specifică a tipului semafor este dată printr-un număr natural (întreg pozitiv) și o coadă atașată operației `WAIT(SEMAFOR)`, dar despre disciplina de coadă nu se precizează nimic [4].

Dacă valorile pe care le poate lua o variabilă semafor sunt numai 0 și 1, atunci semaforul este unul binar. Cele două tipuri de date, ***semafoarele generale*** și ***semafoarele binare***, sunt echivalente [4].

Definițiile operațiilor cu semafoare sunt următoarele:

- `WAIT(SEMAFOR)`:
  dacă `SEMAFOR > 0`
  atunci `SEMAFOR ← SEMAFOR - 1`
  altfel *blocare proces*
- `SIGNAL(SEMAFOR)`:
  dacă există *procese blocate*
  atunci *deblocare un proces*
  altfel `SEMAFOR ← SEMAFOR + 1`

Dijkstra a notat operația `WAIT()` cu `P(SEMAFOR)` și `SIGNAL()` cu `V(SEMAFOR)`, unde `P` și `V` sunt inițialele cuvintelor olandeze: *passeren* (a trece) și, respectiv, *vrygeven* (a elibera). De aici este imediată semnificația celor două operații: operația `WAIT()` permite trecerea procesului pe la semafor, dacă valoarea semaforului este nenulă și, respectiv, blocarea procesului la acel semafor, dacă valoarea lui este zero.

Pentru alegerea procesului care va fi deblocat de operația `SIGNAL()` programatorul poate implementa propria politică de deblocare. Aceasta poate fi aleatoare sau poate respecta mecanismul de lucru cu structuri de tip coadă, coadă cu prioritate ș.a.m.d. Evident, alegerea aleatoare poate amâna la nesfârșit deblocarea unui anumit proces, afectând vivacitatea sistemului.

Realizarea excluderii reciproce cu ajutorul semafoarelor

Fiecare dintre operațiile `WAIT(SEMAFOR)` și `SIGNAL (SEMAFOR)` este o operație atomică, ceea ce permite semafoarelor să fie utilizate cu succes în rezolvarea problemei excluderii mutuale. Concret, atomicitatea operației `WAIT (SEMAFOR)` înseamnă că succesiunea de operații interne *comparare* + *decrementare* este indivizibilă. De asemenea, pentru operația `SIGNAL(SEMAFOR)` succesiunile *comparare* + *deblocare* sau *comparare* + *incrementare* sunt indivizibile.

În timp ce două operații asupra aceluiași semafor se execută prin excludere reciprocă, este posibilă suprapunerea executării a două operații asupra unor semafoare diferite.

Algoritmul de rezolvare a problemei excluderii reciproce folosind un singur semafor `sem` este următorul:

```
repetă la infinit
    WAIT(sem); // protocol de intrare în secțiunea critică
                  execută secțiune critică;
    SIGNAL(sem); // protocol de ieșire din secțiunea critică
                    execută secvență necritică;
sfârșit repetă
```

Conform acestui algoritm, prin apelul primitivei `WAIT()` procesul își anunță intenția de a intra în secțiunea critică, iar prin `SIGNAL()` semnalează semaforului terminarea executării secțiunii critice.

Pentru ca semaforul `sem` să rezolve problema excluderii mutuale, valoarea sa inițială trebuie să fie 1. [4]

Exemple de utilizare a semafoarelor

Utilizarea semafoarelor este o modalitate de lucru folosită cu succes în rezolvarea excluderii mutuale a proceselor concurente, în sincronizarea proceselor cooperative, în instanțierea multiplă a unei resurse partajate.

Exemple

Fie o intersecție semaforizată a unor artere de circulație rutieră. În limbajul sistemelor concurente se poate spune că intersecția este o resursă critică partajată de procesele-mașini care încearcă să treacă prin intersecție, adică să folosească în comun, eventual chiar simultan, resursa critică-intersecție. Rolul semaforului din intersecție este *jucat* în programarea sistemului concurent de o variabilă de tip `SEMAFOR`. Este vorba de un semafor binar, deoarece el poate avea numai valorile 0 sau 1, respectiv *roșu* sau *verde*.

În acest caz procesele-mașini își anunță intenția de a intra în intersecție printr-o operație `WAIT(SEMAFOR)`. De asemenea, mașinile testează periodic starea semaforului prin apeluri `SIGNAL(SEMAFOR)` pentru a decide când au dreptul să intre în intersecție. Cât timp un semafor din intersecție este verde, nici un alt semafor din aceeași direcție de trafic nu poate fi verde, dar un semafor din altă direcție poate fi verde. Această situație practică asigură accesul exclusiv al mașinilor în intersecție, respectiv al proceselor la utilizarea resursei critice.

Atunci când o mașină încearcă să treacă prin intersecție - adică un proces execută o operație `WAIT(SEMAFOR)` - și găsește semaforul roșu (variabila `SEMAFOR` are valoarea 0) acea mașină va fi blocată la semafor. În continuare, mașina intenționează să plece de la semafor (procesul trebuie să fie deblocat). Pentru aceasta procesul va executa periodic operații `SIGNAL(SEMAFOR)` până în momentul în care semaforul va fi verde (variabila `SEMAFOR` va avea valoarea 1).

Dacă în momentul în care o mașină încearcă să treacă prin intersecție (un proces execută o operație WAIT (SEMAFOR)) și găsește semaforul verde (adică variabila SEMAFOR are valoarea 1) atunci acea mașină va fi liberă să intre în intersecție.

Problema producător-consumator poate fi rezolvată cu două semafoare cu următoarele semnificații: unul numit DATE, a cărui valoare să reprezinte datele din *buffer* la un moment dat (numărul articolelor produse și încă neconsumate) și celălalt numit SPAȚII, a cărui valoare să reprezinte locațiile libere din *buffer* la un moment dat. Valoarea inițială este 0 pentru DATE și numărul total de locații ale *buffer*-ului pentru SPAȚII.

Limitele semafoarelor. Perspective

Dacă presupunem că semaforul este singurul mecanism de tip *IPC* al unui limbaj de programare, atunci pot apărea o serie de probleme, cum ar fi:

- se poate greși ușor în programarea semafoarelor: dacă programatorul uită să folosească apelul WAIT atunci, accidental, poate să acceseze date partajate neprotejate; sau dacă programatorul uită să execute un apel SIGNAL, atunci poate să lase o structură de date blocată indefinit;
- nu este posibil să avem o listă de semafoare ca argument pentru WAIT; dacă această posibilitate ar fi fost acceptată, atunci ar putea fi programate diferite variante de ordonare a acțiunilor, conform cu starea curentă a sosirii semnalelor de tip SIGNAL.
- timpul în care un proces este blocat la un semafor nu este limitat;
- nu există nici un mijloc prin care un proces să poată controla un alt proces.

Ținând cont de toate aceste probleme, este evident că un limbaj de programare destinat aplicațiilor concurente trebuie să dețină și mecanisme de tip *IPC* care să completeze facilitățile oferite de semafoare. Multe dintre aceste inconveniente se elimină dacă situațiile de blocare în așteptare sunt asociate cu posibilitatea de a crea procese-fii care să continue execuția în timp ce părintele rămâne în așteptare.

Potrivit lui ***Bacon*** [2], alternativele pentru semafoare sunt tipurile de date EVENTCOUNT și SEQUENCER, precum și bibliotecile de lucru cu *thread*-uri cum ar fi *thread*-urile *POSIX*. În plus, putem aminti și tipurile abstracte de date mai evoluate cum este *monitorul*.

Monitoare

Un ***monitor*** are structura unui obiect abstract de date. Datele încapsulate într-un monitor sunt partajate și fiecare operație este executată sub excludere mutuală. Implementarea monitoarelor trebuie să asigure că, în fiecare moment, numai un proces este activ în monitorul respectiv. Monitoarele au avantajul de a asigura excluderea mutuală la nivelul operațiilor interne dar, dacă se impune sincronizarea pe condiție, monitoarele pot folosi și *variabile de condiție*.

Monitoarele corespund modelului de calcul paralel bazat pe memorie partajată. Concret, datele unui monitor sunt partajate (publice).

Conform modelului introdus de ***Hoare*** în [8], un monitor are structura unui obiect de date abstract, care încapsulează date și operații. Datele încapsulate într-un monitor sunt partajabile, iar fiecare operație este executată sub excludere mutuală. Din aceste considerente rezultă imediat trei avantaje importante ale monitoarelor și anume:

- monitoarele introduc posibilitatea modularizării aplicațiilor concurente;
- executarea unei secțiuni critice la nivelul unui proces se poate lansa prin apelul unei metode a unui monitor;
- monitoarele rezolvă prin definiția lor problema excluderii mutuale a accesului la secțiunile critice.

Pe lângă asigurarea excluderii mutuale, pentru rezolvarea problemelor care apar în aplicațiile concurente, trebuie completate mecanismele pentru sincronizarea pe condiție (sincronizarea condiționată). În limbajele care permit efectuarea de operații cu monitoare, această problemă este rezolvată prin introducerea unui tip de date special pentru *variabilele de condiție*. Programatorul declară o variabilă de condiție necesară aplicației și modulul de gestionare a monitoarelor va defini o coadă asociată variabilei respective. Mai mult, gestionarea monitoarelor se referă implicit și la gestionarea acestor cozi astfel încât, împreună cu protocolul de organizare a proceselor care solicită accesul la monitor, să se asigure sincronizarea proceselor cu respectarea condițiilor asociate variabilelor definite.

Asupra variabilelor de condiție pot fi efectuate operațiile WAIT și SIGNAL cu următoarele semnificații: WAIT determină autoîntârzierea procesului care apelează WAIT (VAR_COND) prin depunerea lui în coada asociată variabilei VAR_COND; ulterior, acest proces este eliberat de către un alt proces care execută o operație SIGNAL(VAR_COND) asupra aceleiași variabile de condiție. Dacă există mai multe procese care așteaptă la VAR_COND atunci unul va fi cu siguranță eliberat. Implicit, SIGNAL nu are nici un efect în cazul în care coada asociată variabilei VAR_COND este vidă.

Un semafor se reprezintă printr-un întreg și o coadă, în timp ce o variabilă de condiție se reprezintă numai printr-o coadă de așteptare. De aici rezultă două îmbunătățiri ale limitelor semafoarelor:

- nu se pune problema semnalelor SIGNAL de tip "*wake-up waiting*";
- procesul care execută WAIT este depus necondiționat în coada de așteptare a variabilei de condiție.

Monitoarele sunt structuri pasive care permit proceselor active care le apelează metodele să se succeadă pentru a obține cooperarea și/sau competiția cu alte procese. Prin mecanisme specifice, monitoarele organizează cozi pentru resursele partajate și gestionează *buffer*-e de comunicare între diferite tipuri particulare de procese active (de exemplu, producător - consumator, cititor - scriitor).

Monitoarele reprezintă modelul de sincronizare folosit în *Java* [9].

Comunicarea prin mesaje

În paragrafele anterioare am luat în discuție semafoarele (ca mecanisme *low-level* pentru implementarea concurenței) și monitoarele (ca exemple de obiecte procedurale, pasive care se implementează folosind partajarea memoriei).

În această secțiune introducem mecanisme de rezolvare a comunicării datelor și sincronizării proceselor care se implementează *fără partajarea memoriei*.

Abordarea cea mai generală a primitivelor de limbaj fără partajarea memoriei presupune atât sincronizare cât și transfer de date. Concret, avem două modalități de exploatare: flux de date și transfer de mesaje.

Mecanismul de lucru cu un ***flux de date*** presupune că un proces poate trimite octeți într-un flux nestructurat și un alt proces poate citi octeți de pe fluxul respectiv. În acest caz, pe flux nu sunt prezente informații despre structura octeților sau despre tipul de date transferate pe flux. O aplicație care folosește un astfel de mecanism trebuie să poată interpreta datele de pe flux conform cu o structură definită anterior, în timp ce sistemul însuși nu cunoaște această structură și nici nu oferă suport pentru aceasta.

Transferul de mesaje este un mecanism de lucru puternic, înrudit cu transferul informațiilor care însoțește transmiterea parametrilor la apelul de procedură. Mesajul conține un antet și un corp, fiecare având funcții bine delimitate în structura mesajului. Astfel, antetul (*header*) conține destinația informației conținute de mesaj, iar corpul mesajului (*body*) conține informația propriu-zisă și argumentele de transfer. Aceste argumente pot să fie supuse sau nu unor operații de validare a mesajului care se pot desfășura în mai multe etape.

Programele concurente care angajează transfer de mesaje sunt numite ***programe distribuite***. [7]

În funcție de tipul de sincronizare folosit pentru comunicarea datelor între procese prin mesaje avem:

- comunicare asincronă - procesul expeditor nu se interesează dacă mesajul transmis este recepționat de destinatar, cu atât mai puțin dacă este recepționat corect; acest model poate fi comparat cu comunicarea prin poștă;
- comunicare sincronă - după ce a transmis un mesaj, procesul expeditor se blochează în așteptarea confirmării de primire de la expeditor; acest model poate fi comparat cu comunicarea prin fax sau prin telefon.

Aici, prin *expeditor* și *destinatar* ne referim la procesele care comunică printr-un mesaj oarecare, adică la procesul care trimite mesajul, respectiv la procesul care recepționează mesajul (procesul căruia îi este adresat mesajul).

Se poate observa că, în cazul comunicării sincrone expeditorul și destinatarul trebuie să-și dedice un interval de timp exclusiv acestei comunicări, trebuie *să se întâlnească* pentru a comunica, respectiv să se sincronizeze pentru predarea (preluarea) mesajului. Spunem că are loc o întâlnire (un *rendez-vous*) între procesele care comunică. Acesta este mecanismul de sincronizare simplă a comunicării.

Dacă, în plus, comunicarea este prevăzută astfel încât expeditorul să primească împreună cu confirmarea de recepție și un mesaj de răspuns din partea destinatarului, atunci are loc un *rendez-vous extins* sau *invocare la distanță*.

În continuare ne propunem să luăm în discuție transmiterea asincronă și transmiterea sincronă de tip *rendez-vous* a mesajelor, urmând ca în paragraful următor să abordăm invocarea la distanță.

Transmiterea asincronă de mesaje, transmiterea sincronă de mesaje, precum și primitivele pentru invocare la distanță sunt mecanisme echivalente pentru comunicarea datelor și sincronizarea proceselor în sisteme concurente. [4]

Transmiterea asincronă de mesaje

Modalitatea asincronă de transmitere a mesajelor este varianta generală, care nu impune restricții de timp asupra comunicării între procesele care comunică prin transfer de mesaje. După cum am afirmat anterior, structura tipică a unui mesaj constă dintr-un antet și corpul mesajului. La rândul lui, antetul este format din informații referitoare la destinatarul mesajului, informații referitoare la expeditorul mesajului și informații referitoare la tipul mesajului.

Pentru ca transferul mesajelor să fie funcțional, în sistem trebuie să existe un serviciu direct răspunzător de lucrul cu mesaje. Acest serviciu de transport al mesajelor folosește antetul fiecărui mesaj pentru a identifica tipul mesajului și procesele care comunică. În plus, acest serviciu poate stabili și urmări respectarea unor protocoale de comunicare între procese.

Primitivele specifice de lucru pentru transmiterea de mesaje sunt grupate în perechile: `SEND` și `WAIT`, respectiv `SEND` și `RECEIVE`.

Dacă avem două procese `A` și `B`, care comunică la un moment dat prin transmitere de mesaje, atunci acțiunile celor două procese sunt următoarele:

- procesul `A` construiește și configurează mesajul, după care execută o operație `SEND(destinatar, mesaj);`
- procesul `B` așteaptă la punctul de întâlnire cu `A` executând operații `WAIT/RECEIVE(expeditor, spatiu_adresa)`, unde al doilea parametru reprezintă adresa de memorie la care trebuie să se depună mesajul.

Transmiterea este asincronă deoarece procesul `A` nu așteaptă de la procesul `B` vreun semnal prin care să i se confime recepționarea mesajului.

Acest scenariu presupune că procesele care comunică își cunosc reciproc identitatea (pot să se numească reciproc prin identificatorii transmiși ca parametri) și, în plus, există un protocol de transmitere de mesaje pe baza căruia expeditorul construiește mesajul de transmis și destinatarul interpretează mesajul primit.

În general, recepționarea mesajului coincide cu operația de copiere a mesajului din spațiul de adresă al expeditorului `A` în spațiul de adresă al destinatarului `B`, eventual prin intermediul unui *buffer* de colectare a mesajelor adresate lui `B` care este gestionat de serviciul de transmitere de mesaje. Astfel rezultă că, pentru transmiterea mesajului este necesar să se creeze două copii ale acestuia.

Rezolvarea și implementarea transmiterii asincrone de mesaje a condus la definirea unor noțiuni și concepte noi, cum ar fi: comunicarea prin porturi și canale, comunicare de tip *broadcast* (1 *la n*, sau *one-to-everyone*), *multicast* (1 la fiecare dintr-un grup de receptori înscriși) sau *unicast* (1 *la* 1) [2], [4], [7]. Implicit, aceste concepte se regăsesc în diverse primitive la nivelul fiecărui limbaj de programare care le recunoaște.

În interiorul unui modul de program, un mesaj poate fi transmis la o anumită locație (de exemplu, către o variabilă locală a programului respectiv). Dacă această locație este destinată comunicării programului cu exteriorul, atunci limbajul programului poate să o identifice ca pe un ***port de ieșire*** sau un ***canal de comunicare***. În timp ce aceste entități pot fi stabilite la începutul programului, locația care preia mesajul la intrarea în program, adică ***portul de intrare***, se poate stabili numai în timpul configurării mesajului. În plus, la configurare se stabilește care este procesul asociat unui anumit port de ieșire, respectiv care este procesul care va prelua datele de pe un anumit canal.

Transmiterea sincronă de mesaje

Reluând mecanismul de transmitere de mesaje între procesele `A` și `B` descris anterior, transmiterea este sincronă dacă procesul expeditor `A` este întârziat până când destinatarul execută operația `WAIT` prin care confirmă disponibilitatea de a recepționa mesajul transmis de `A`.

Față de operațiile interne executate asupra mesajului la transmiterea asincronă, mesajul este copiat numai o singură dată, atunci când expeditorul și destinatarul s-au sincronizat (s-au întâlnit) și pot comunica. Deoarece condiția de sincronizare este implicită, rezultă că sistemele cu transmitere sincronă de mesaje sunt mai ușor de gestionat decât cele asincrone.

Cele două variante de transmitere sincronă de mesaje sunt:

- transmiterea cu sincronizare simplă (*rendez-vous*);
- invocarea la distanță.

La rândul ei, invocarea la distanță poate fi reprezentată de un *rendez-vous extins* sau de *apelul de procedură la distanță* (*RPC*).

Prin *rendez-vous*, operațiile de transmitere și preluare sunt sincronizate, în sensul că transmiterea are loc efectiv numai dacă expeditorul este pregătit să transmită și destinatarul este pregătit să recepționeze. În caz contrar, procesul care ajunge primul la "întâlnire" este suspendat în așteptarea celuilalt. Cu alte cuvinte, nici un proces nu poate avansa înainte de terminarea transmiterii.

Invocarea la distanță

Ambele metode ale invocării la distanță, atât *rendez-vous-ul extins*, cât și *RPC*, combină aspecte întâlnite la monitoare și la transmiterea sincronă de mesaje. Ca și la monitoare, un modul sau un proces exportă operații care sunt invocate printr-un apel de tip *call*. La fel ca în cazul unei secvențe `SEND - WAIT` care este specifică transmiterii sincrone de mesaje, execuția unui apel *call* întârzie procesul care îl execută. [7]

Noutatea pe care o aduc mecanismele *RPC* și *rendez-vous extins* constă în faptul că o operație de transmitere poate fi asociată unui canal de comunicație bidirecțional: atât de la apelant la procesul care oferă serviciul solicitat de apel, cât și invers. Apelantul întârzie până când primește un răspuns legat de încheierea execuției operației solicitate.

Diferența dintre *RPC* și *rendez-vous extins* constă în modul în care sunt deservite cererile invocate. Astfel, o primă abordare este să se declare o procedură pentru fiecare operație și să se creeze un proces nou pentru fiecare apel nou. Această abordare se numește ***RPC*** (*Remote Procedure Call*) deoarece apelantul și corpul procedurii apelate pot să se găsească pe mașini diferite (la distanță). O a doua abordare este să se asigure faptul că procesul apelant "se întâlnește" (stabilește un *rendez-vous*) cu un proces existent. Un *rendez-vous extins* constă din trei elemente:

- o operație de *intrare*, `ACCEPT` - prin care un proces așteaptă să fie invocat;
- o operație de *prelucrare a cererii*;
- o operație de *furnizare a rezultatului* (răspunsului).

În cazul comunicării prin apel de procedură, *mesajul* constă în valorile parametrilor efectivi transmiși entității apelate, iar *răspunsul* este conținut de valorile parametrilor returnați prin referință entității apelante.

Dezvoltarea programării concurente se poate face folosind procese și monitoare, care partajează același spațiu de adrese. Într-un astfel de model, monitoarele încapsulează variabile partajate, în timp ce procesele comunică și se sincronizează prin apelarea metodelor monitoarelor. Cu mecanismul *RPC* se poate folosi o singură componentă de programare, un modul, care să conțină atât procesele cât și procedurile (metodele). De asemenea, se permite modulelor să fie rezidente în spații de adresă diferite, eventual chiar în noduri diferite ale rețelei. Procesele interne ale unui modul pot partaja variabile și pot apela proceduri declarate în acel modul. Totuși, procesele dintr-un modul pot comunica cu procesele dintr-un alt modul numai prin apeluri de proceduri.

În esență, *RPC* furnizează numai un mecanism de comunicare între module. În interiorul unui modul trebuie programată explicit sincronizarea proceselor. În plus, programatorul trebuie să creeze procese pentru manipularea datelor comunicate prin *RPC*.

Mecanismul *rendez-vous* combină acțiunea de deservire a unei cereri cu alte prelucrări solicitate de apel. Cu mecanismul *rendez-vous*, un proces exportă operații (în sensul că operațiile sunt vizibile din afară), care pot fi apelate de alte procese.

Ca și în cazul mecanismului *RPC*, un proces invocă o operație prin intermediul unui apel *call*. Pentru *RPC*, apelul *call* numește un alt proces care să execute și o operație din acel proces care să se execute. Spre deosebire de *RPC*, în cazul mecanismului *rendez-vous*, o operație este executată de același proces care o exportă.

Pentru a pune în evidență deosebirile dintre mecanismele folosite pentru transmiterea de mesaje, se pot face trei observații referitoare la comunicarea prin canal și invocarea la distanță:

- comunicarea prin canal este identificată cu transmiterea de mesaje între procesele care comunică, în timp ce invocarea la distanță constă în apelarea din cadrul unuia dintre procese a unei secvențe de instrucțiuni a celuilalt proces, urmând ca aceste instrucțiuni să se execute la punctul de întâlnire;
- în cazul comunicării prin canal, sursa și destinația sunt univoc determinate, în timp ce pentru invocarea la distanță este precizată numai destinația. De aici rezultă că invocarea la distanță permite stabilirea unor legături de comunicare de tip *n* la 1;
- prin canal informația circulă de la sursă la destinație, în timp ce invocarea la distanță permite transferuri de mesaje în ambele sensuri.

Comparând comunicarea cu precizarea indirectă a partenerilor și invocarea la distanță, putem spune că invocarea la distanță permite interacțiunea proceselor implicate ca entități active în comunicare, în timp ce comunicarea prin canal sau prin resurse trebuie să apeleze la entități intermediare, pasive.

Totuși, comunicarea prin canale și invocarea la distanță sunt mecanisme echivalente de realizare a aplicațiilor concurente [4].

În concluzie, putem da o altă clasificare, mai generală, utilă pentru prezentarea mecanismelor de comunicare între procese (*IPC*) și anume:

- low-level: semafoare
- memorie partajată
  - ♦ secțiuni critice
  - ♦ expresii de tip cale
  - ♦ mecanisme procedurale
    - ▪ pentru obiecte pasive: monitoare
    - ▪ pentru obiecte active: rendez-vous procedural
- fără memorie partajată = comunicare prin mesaje
  - ♦ asincron = cu buffer intermediar
    - ▪ porturi și canale
    - ▪ broadcast și multicast
  - ♦ sincron
    - ▪ sincronizare simplă = rendez-vous
    - ▪ invocare la distanță
      - ❖ rendez-vous extins
      - ❖ RPC

Pentru entitățile din această clasificare, netratate în acest serial se pot consulta [2], [3], [4], [6].

Bibliografie

1. ***Coulouris G.F. ș.a.***, *Distributed systems - concepts and design*, Addison-Wesley, Wokingham, 1999
2. ***Bacon J.***, *Concurrent Systems - operating systems, database and distributed systems: an integrated approach*, Addison-Wesley, 1998
3. ***Vasilescu A.***, *Sisteme concurente, Referat în cadrul stagiului de pregătire pentru doctorat*, Universitatea Babeș-Bolyai, Cluj-Napoca, 2001
4. ***Georgescu H.***, *Programare concurentă - teorie și aplicații*, Editura Tehnică, București, 1996
5. ***Dijkstra E.W.***, *The structure of THE operating system*, Comm. ACM 11(5), 1968
6. ***Boian F.M.***, *Programarea distribuită în Internet - metode și aplicații*, Editura Albastră, Cluj-Napoca, 2000
7. ***Andrews G.R.***, *Concurrent Programming - Principles and Practice*, Benjamin/Cummings, Redwood City CA, 1991
8. ***Hoare C.A.R.***, *Monitors: an operating system structuring concept*, Comm. ACM 17(10), 1974
9. ***Scheiber E.***, *Lecții de calcul paralel*, Reprografia Universității "Transilvania" Brașov, 1999

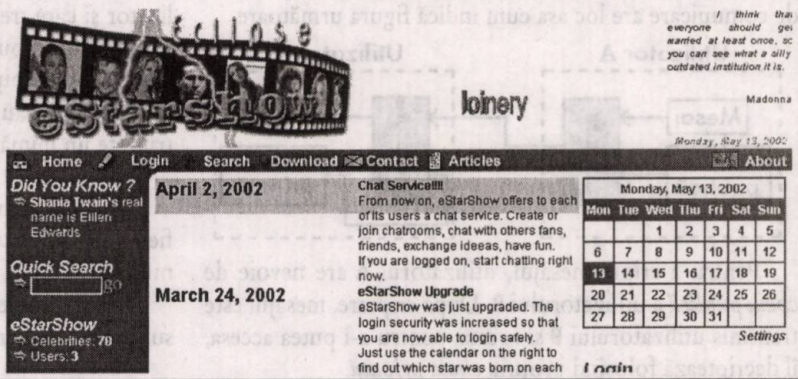
D-na Anca Vasilescu este asistent universitar la Universitatea Transilvania din Brașov și poate fi contactată prin e-mail la adresa **vasilex@info.unitbv.ro**. *D-na Oana Georgescu este asistent universitar la Universitatea Sextil Pușcariu din Brașov și poate fi contactată prin e-mail la adresa* **o.georgescu@info.unitbv.ro**.

www.estarshow.go.ro

Welcome to eStarShow!

Here you can find information about your favorite music and movie stars, information about movies, albums, etc.

- *bibliographies*
- *discographies*
- *filmographies*
- *pictures*
- *quotes*
- *links*



Tehnici de CRIPTARE

Claudiu Soroiu

În acest ultim episod al serialului dedicat tehnicilor de criptare vom prezenta cel mai cunoscut sistem de codificare cu cheie publică a mesajelor, și anume RSA; acesta este utilizat cu succes de foarte mulți ani.

Introducere

Algoritmii de criptare a mesajelor utilizând *chei publice* se folosesc atunci când un număr mare *n* de utilizatori trebuie să comunice între ei (*fiecare cu fiecare*). În cazul în care aceștia ar folosi tehnici tradiționale de criptare, atunci ar avea nevoie de $n \cdot (n - 1) / 2$ chei, spre deosebire de algoritmii de criptare cu *chei publice* care folosesc doar *n* chei.

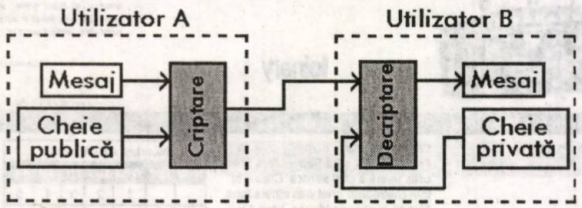
În sistemele de criptare cu *cheie publică*, dacă un utilizator dorește să comunice cu alții, trebuie să își construiască două chei:

- o *cheie publică* pe care o va transmite oricărui alt utilizator care dorește să-i trimită un mesaj criptat;
- o *cheie privată* care este secretă și pe care o va folosi la decriptarea unui mesaj criptat cu *cheia publică*.

Cele două chei se presupun a fi unice pentru fiecare utilizator.

Secretul sistemelelor de criptare cu *cheie publică* este acela că dacă un utilizator deține *cheia publică* a unui alt utilizator ale cărui mesaje dorește să le intercepteze și decripteze, de obicei, este imposibil să construiască *cheia privată* și să o folosească pentru decriptarea mesajelor. În cazul în care *cheia privată* se poate calcula pe baza *cheii publice*, atunci timpul necesar unui sistem de calcul este suficient de mare și, după decriptarea mesajului acesta nu ar mai fi de actualitate.

Dacă avem doi utilizatori **A** și **B**, și utilizatorul **A** dorește să-i trimită utilizatorului **B** un mesaj criptat, procesul de comunicare are loc așa cum indică figura următoare.



Pentru a cripta mesajul, utilizatorul **A** are nevoie de *cheia publică* a utilizatorului **B**. După criptare, mesajul este transmis utilizatorului **B** și acesta, pentru a-l putea accesa, îl decriptează folosind propria *cheie privată*.

Presupunând că avem un grup de *n* utilizatori care au nevoie să comunice între ei în siguranță și toți folosesc o aceeași tehnică de criptare cu *cheie publică*, atunci, pentru comunicare, fiecare utilizator *i* din cei *n* are nevoie de:

- *cheile publice* ale celorlalți *n* - 1 utilizatori; o *cheie publică* a unui utilizator *j* este folosită în procesul de criptare a unui mesaj de către utilizatorul *i*, urmând ca acesta să fie trimis utilizatorului *j*;
- o *cheie privată* care este folosită de către utilizatorul *i* pentru a decripta orice mesaj care sosește de la unul dintre ceilalți și a fost criptat folosindu-se *cheia publică* a utilizatorului *i*.

RSA

RSA este un sistem de criptare cu *cheie publică* dezvoltat în anul 1977 de către profesorii de la *MIT* (*Massachusetts Institute of Technology*) *Ronald L. Rivest*, *Adi Shamir* și *Leonard M. Adleman*, cu scopul de a asigura securitatea datelor pe *Internet*.

Generarea cheilor

Algoritmul *RSA* are la bază elemente de teoria numerelor și teoria grupurilor.

Primul pas în folosirea algoritmilor de criptare și decriptare pentru tehnica *RSA* constă în construirea celor două chei.

Pe baza unei parole (șir de caractere) introdusă de utilizator și care trebuie să aibă o lungime suficient de mare, se generează două numere prime mari (cu o lungime mai mare de 64 de biți) *p* și *q*.

După ce s-au generat cele două numere prime se construiește un număr *n* ca fiind produsul dintre *p* și *q*.

Fie $\phi = (p - 1) \cdot (q - 1)$.

La pasul următor se caută un număr *e* care trebuie să fie relativ prim cu ϕ, adică cel mai mare divizor comun al numerelor *e* și ϕ este 1.

În continuare se construiește numărul *d* ca fiind inversul numărului *e* modulo ϕ:

$$d = \texttt{rest}([e^{-1}/\phi])$$

Numerele n și e formează *cheia privată*, în timp ce *cheia publică* este dată de numerele n și d.

Pentru ca generarea cheilor să fie optimă în timp, la calcularea numărului d se folosește forma extinsă a algoritmului lui *Euclid*.

Algoritmul extins al lui Euclid

După cum se cunoaște, algorimul lui *Euclid* calculează cel mai mare divizor comun a două numere naturale astfel:

```
algoritm Euclid(a, b)
dacă b = 0
  atunci returnează a
  altfel returnează Euclid(b, rest([a/b]))
```

Forma extinsă a algoritmului lui *Euclid*, pe lângă cel mai mare divizor comun d a două numere a și b, determină două numere întregi x și y cu proprietatea $d = a \cdot x + b \cdot y$. Se poate demonstra matematic că perechea de numere x și y este unică.

Forma extinsă a algoritmului lui *Euclid* este următoarea:

```
algoritm EuclidExtins(a, b)
dacă b = 0
  atunci returnează (a, 1, 0)
  altfel
    (d', x', y') ← EuclidExtins(b, rest([a/b]))
    returnează (d', y', x' - [a/b] · y')
```

Revenind la determinarea inversului numărului e, se poate demonstra că pentru orice număr pozitiv ϕ relativ prim cu e, ecuația $\text{rest}([(e \cdot x)/\phi]) = 1$ are soluție unică și nu are soluție în cazul în care nu sunt respectate condițiile. În acest caz avem:

$\text{EuclidExtins}(e, \phi) = (1, x, y)$

În cadrul clasei de resturi modulo ϕ, avem $\phi \cdot y = 0$, de unde rezultă că x este inversul numărului e.

Criptarea și decriptarea

Fie m mesajul (un număr) care trebuie să fie criptat cu lungimea mai mică decât n.

Relația prin care se obține mesajul criptat c, din mesajul m este:

$$c = \text{rest}([m^e/n])$$

După cum am afirmat anterior, la decriptarea mesajului c se vor folosi numerele d și n astfel:

$$m = \text{rest}([c^d/n])$$

Folosind aceste relații se poate demonstra matematic corectitudinea algoritmului *RSA*.

Pentru ca procesele de criptare și decriptare să fie optime în timp, se folosesc diverși algoritmi performanți pentru calcularea restului.

Unul dintre cei mai utilizați astfel de algoritmi se bazează pe *teorema chineză a restului*. Mai multe detalii despre această teoremă puteți găsi în cartea ***Introducere în algoritmi***, scrisă de *Thomas H. Cormen*, *Charles E. Leiserson* și *Ronald R. Rivest*, a cărei traducere în limba română a apărut la editura *Computer Libris Agora* din *Cluj-Napoca*.

Securitatea RSA

Securitatea sistemului este asigurată de faptul că este foarte dificilă factorizarea numerelor întregi mari. În cazul în care cineva reușește să factorizeze numărul n, atunci poate obține *cheia privată* din *cheia publică*.

În cazul în care factorizarea numerelor mari ar fi o operație simplă, un sistem de securitate bazat pe algoritmul *RSA* ar fi foarte ușor de evitat.

Pentru a asigura o securitate ridicată a datelor utilizând *RSA*, este indicat să se folosească numere care au suficient de multe cifre cât să îngreuneze foarte mult factorizarea dar criptarea mesajelor să fie optimă în timp.

În unele cazuri, pentru o securitate mai mare, se folosește o criptare hibridă a mesajelor cu ajutorul algoritmului *RSA* și a unui alt algoritm rapid cu cheie nepublică, iar în alte cazuri se folosește o criptare dublă *RSA* și, în consecință, fiecare dintre cele două chei este formată din patru elemente: (n_1, n_2, e_1, e_2) și (n_1, n_2, d_1, d_2).

Semnături digitale

Cu toate că părerile multora sunt contrare, *semnătura publică* (digitală) reprezintă cea mai importantă utilizare a sistemului *RSA*.

O *semnătură cu cheie publică* îi permite destinatarului să verifice autenticitatea unui mesaj. O astfel de semnătură trebuie să aibă următoarele proprietăți:

- semnătura nu trebuie să poată fi falsificată;
- semnătura nu trebuie să fie reutilizabilă; semnătura reprezintă o funcție a documentului și nu poate fi transferată la un alt document.

Sistemul *RSA* de *semnare digitală a mesajelor* are aceste proprietăți.

Algoritmul *RSA* de semnare digitală a mesajelor este aproape identic cu cel folosit pentru criptarea cu *cheie publică*, singura diferență fiind aceea că rolul cheilor se schimbă, adică perechea de numere n și d constituie *cheia privată*, iar perechea de numere n și e constituie *cheia publică*.

Fie m un mesaj care urmează să fie autentificat.

Dacă un utilizator **A** dorește să trimită mesajul m autentificat unui utilizator **B**, atunci **A** păstrează cheia *privată* (n, d) și trimite cheia publică (n, e) și mesajul autentificat c, unde:

$$c = \text{rest}([m^d/n])$$

Pentru a valida semnătura, utilizatorul **B** verifică dacă:

$$m = \text{rest}([c^e/n])$$

În continuare, mesajul original poate fi preluat doar dacă expeditorul deține *cheia privată*. În consecință, semnătura este autentică.

Claudiu Soroiu este redactor al GInfo. Poate fi contactat prin e-mail la adresa **csoroiu@yahoo.com**.

În numărul URMĂTOR

După vacanța de vară GInfo va reveni la sfârșitul lunii octombrie. Câteva dintre articolele pe care le veți putea citi sunt prezentate pe scurt în continuare...

ONI 2002

La pagina 9 a acestui număr ați putut citi enunțurile problemelor propuse spre rezolvare la ediția din acest an a Olimpiadei Naționale de Informatică. Suntem siguri că mulți dintre cititori sunt interesați de soluțiile acestor probleme. În numărul următor vom prezenta modalitățile de rezolvare a câtorva dintre cele 24 de probleme de la ONI 2002.

Bursele Agora

În numărul următor veți găsi enunțurile problemelor propuse spre rezolvare la faza finală a celei de-a treia ediții a concursului de programare organizat de revista noastră. De asemenea, veți găsi diverse informații referitoare la desfășurarea finalei. De asemenea, va fi prezentat regulamentul de desfășurare a celei de-a patra ediții a concursului Bursele Agora. Veți găsi diverse alte informații referitoare la ediția 2002/2003 a concursului, precum și enunțurile problemelor propuse spre rezolvare la prima rundă.

Vara 2002

În perioada verii vor apărea, cu siguranță, o mulțime de noutăți în diferite domenii ale tehnologiei informației. Numărul 12/6 al revistei noastre va conține un articol care va prezenta succint cele mai importante realizări din diverse domenii cum ar fi hardware, software, jocuri etc. Veți putea citi informații referitoare la o mulțime de noi tipuri de procesoare (Pentium 4 cu frecvența magistralei de 533 MHz, noul Celeron bazat pe același nucleu ca și Pentium 4, viitorul Athlon XP), despre noile plăci grafice produse de nVidia etc. De asemenea, veți găsi câteva detalii despre jocuri mult așteptate cum ar fi WarCraft III sau Age of Mythology. Cu siguranță, vor fi o mulțime de alte noutăți importante; toate vor fi prezentate pe scurt în cadrul acestui articol.

Noul HPQ

® Probabil că mulți dintre cititorii noștri au aflat despre recent finalizata fuziune dintre celebrele companii Hewlett-Packard și Compaq.

i n v e n t

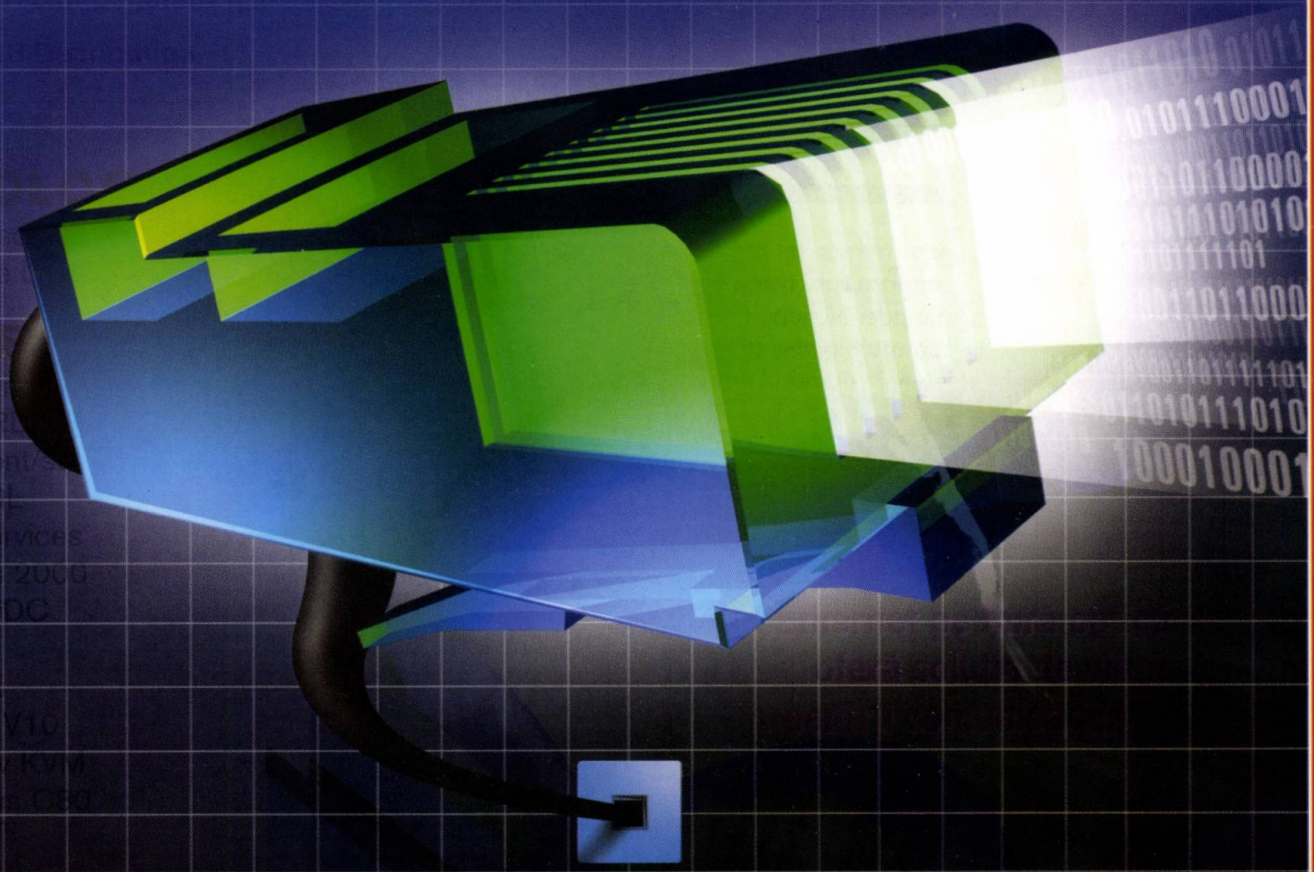
Pentru industria tehnologiei informației, acesta a fost unul dintre cele mai importante și mai neașteptate evenimente care au avut loc în ultimii ani. În numărul următor vom prezenta câteva repere istorice pentru cele două companii care au participat la fuziune, precum și scurta istorie a companiei nou-formate.

COMPAQ

Recunoașterea formelor

Prelucrarea imaginilor este un domeniu foarte interesant care are aplicații diverse în multe ramuri ale științei. În numărul următor va apărea un articol în care se vor prezenta câteva dintre tehnicile de reprezentare și decizie în vederea automatizării procesului de identificare a asemănărilor dintre obiecte sau fenomene. Lucrarea se dorește a fi o introducere în recunoașterea statistică a formelor, prezentând câteva din tehnicile și metodele utilizate în cadrul celor două tipuri de abordări: învățarea supravegheată și nesupravegheată.

NETREPORT



Conectat la viitorul digital

AGORA MEDIA: Str. Tudor Vladimirescu nr. 63/9, Tg. Mureș, CP 230/1, 4300
Tel: 065-166 516, Fax: 065-166 290, e-mail: office@agora.ro

www.netreport.ro

Agora MEDIA

Publicarea acestei reviste este posibilă datorită sponsorizărilor acordate de următoarele firme care s-au decis să sprijine învățământul românesc de informatică

SPONSORI DE PLATINĂ

COMPAQ

SPONSORI DE AUR

FORTE COMPUTERS
Flexibility. Made in Romania

ASTRAL TELECOM

SPONSORI DE ARGINT

sxt
IT SOLUTIONS & SERVICES

hp
invent

ERICSSON

Magua

M.G.T.
EDUCATIONAL

a&C INTERNATIONAL S.A.
the CAD-CAMpany
autodesk
authorized distributor

ALCATEL

*Rugăm companiile care doresc să se alăture acestui efort să ne contacteze prin e-mail la adresa office@ginfo.ro sau la numărul de telefon: 065-166.516

Octombrie 2002
Vol. 12/6

Lei 30.000

Gazeta de informatică

Revista viitorilor specialiști în tehnologia informației

din sumarul numărului:

Bursele Agora A treia ediție a concursului de programare organizat de Gazeta de Informatică

FINALA

Recunoașterea formelor

Lumea pixelilor

Profil Noul HP

Plus

Noi unități CD-RW • Olimpiada Națională de Informatică
Dialog • Concurs de programare • Probleme BOI 2002
Războiul procesoarelor • Teoreme de programare

*a se consuma
înainte de pornirea
calculatorului

Vol. 12, Nr. 6 /2002

gazeta de informatică

Σ sumar

actual.. PAG. 4

Ştiri IT

~ evenimente care au avut ~
~ loc vara aceasta ~

probleme.. PAG. 7

Olimpiada BALCANICĂ de Informatică
~BOI 2002~
~Belgrad, Iugoslavia~

Olimpiada de Informatică a EUROPEI CENTRALE
~CEOI 2002~
~Košice, Slovacia~

Olimpiada INTERNAŢIONALĂ de Informatică
~IOI 2002~
~Yong-In, Coreea de Sud~

problemele propuse spre rezolvare la principalele concursuri internaţionale de programare care au avut loc în această vară

soluţii.. PAG. 22

Olimpiada NAŢIONALĂ de Informatică

~ soluţiile problemelor de la ~
~ clasele a IX-a şi a X-a ~

concurs.. PAG. 30

Etapa finală a concursului "Bursele Agora"

Despre ediţia 2002/2003

serial.. PAG. 36

RECUNOAŞTEREA formelor

Radu-Daniel VATAVU

~ elemente de grafică ~
~ primul episod ~

noţiuni introductive

algoritm simplu de clasificare a formelor

begin.. PAG. 41

TEOREME de programare

Clara IONESCU

~ despre teorie şi practică ~

profil.. PAG. 43

Noul HP

Carmen ŞUŞMAN

~ istoricul companiilor ~
~ HP şi Compaq ~

Redacţia:

Director ştiinţific:
prof. dr. *Horia Georgescu*
Redactor şef:
lect. univ. *Clara Ionescu*
Redactor şef adjunct:
Mihai Scorţaru
Redactori:
prof. *Eugen Ionescu*
Carmen Şuşman
Claudiu Soroiu
Au mai colaborat:
Mihai Stroe (Bucureşti)
Radu-Daniel Vatavu (Suceava)
Tehnoredactare:
Claudiu Soroiu
Coperta şi grafica:
Andrei Paşa

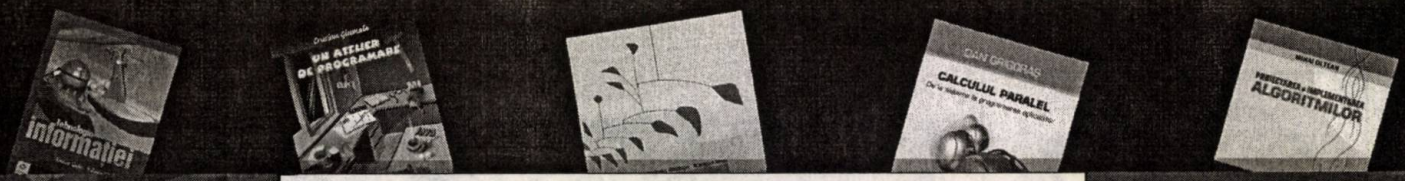
Adresa pentru corespondenţă:
Computer Libris Agora,
Universităţii 8/8,
3400, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 0264/192.422
0265/166.516
Fax: 0265/166.290
http://www.ginfo.ro
ISSN: 1222-7129

Tiparul: *Master Druck* Târgu Mureş
tel: 0265/162.359

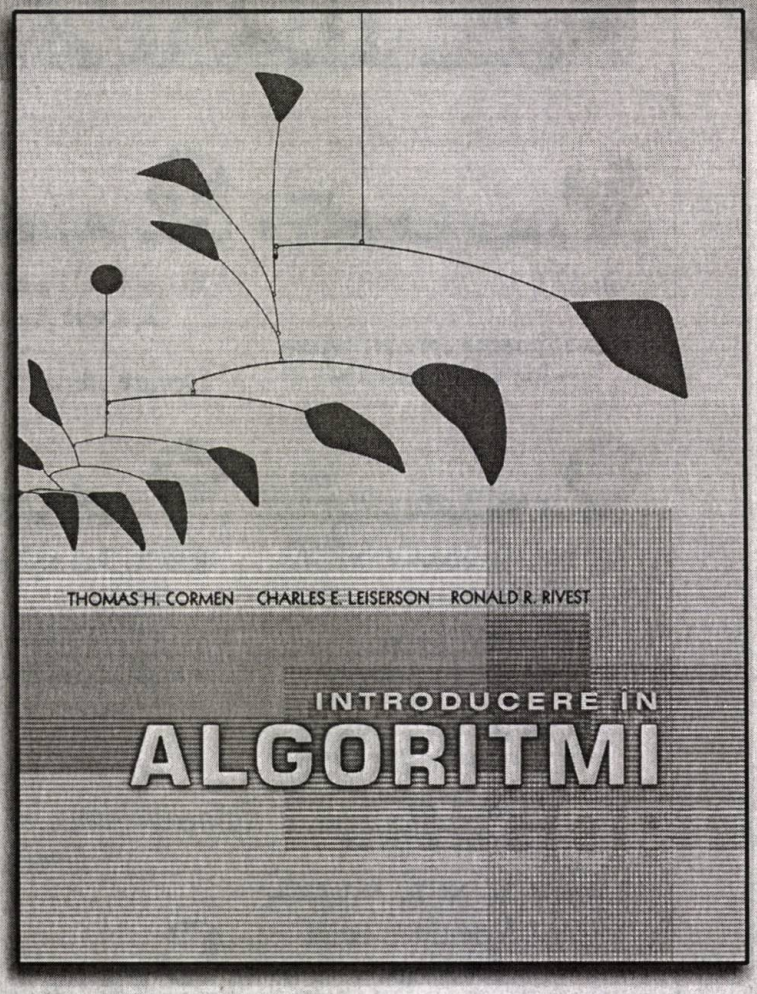
Editor: Agora Media SRL
Agora Media SRL, editor al publicaţiilor "*eWeek România*", "*PC Magazine România*", "*NetReport*" şi "*Gazeta de Informatică*", este membru al Biroului Român pentru Auditarea Tirajelor.
str. T. Vladimirescu nr. 63/9
4300 Târgu-Mureş

Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director tehnic:
Szabó László
Vânzare reclamă:
Mihai Bucuroiu
Cosmin Ursu
Denis Marioara
Marius Comărniceanu

Reprezentanţa Bucureşti:
Director: *Dana Pastramă*
Tel: 021/330.92.81
Fax: 021/330.92.85



COMPUTER LIBRIS AGORA



Cartea se adresează, în primul rând, celor care studiază la nivel elementar sau academic algoritmi și structuri de date. Deoarece tratează aspecte teoretice și practice, legate de proiectarea algoritmilor, este la fel de binevenită elevilor, studenților, profesorilor și autodidacților.
Introducere în algoritmi acoperă atât metodele clasice cât și metodele moderne ca analiza amortizată și algoritmii paraleli.
Prezentarea matematică, riguroasă, este atent detaliată în așa fel încât să fie accesibilă tuturor cititorilor.
Capitolele încep cu elemente de bază, avansând spre subiecte mai complexe.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

La început de an școlar

VALOARE și respect

Scriu acest editorial în data de 11 septembrie și îmi aduc aminte de acea zi tristă a distrugerii, a revoltei, a uluirii. De ce vrea omul, de ce vor oamenii să distrugă? Să distrugă valori realizate de *alții*, valori care au cerut multă muncă, efort și pricepere din partea *altora* și să distrugă în numele unei credințe și a unor principii. De ce? Doar pentru că principiile lor sunt *altfel*, pentru că aceste valori au fost realizate de alții? Cineva ar putea să îmi atragă atenția că realizările respective au conferit putere celui care le-a creat și așadar este firesc ca această putere să fie râvnită de alții. Dar o analiză mult mai subtilă decât una la care mă pricep eu, mai ales într-un editorial pe care îl scriu cu gândul la tineri, ar putea să ajungă și la presupunerea că în momentul în care puterea crește prea ușor, când cel care o deține devine ignorant, atunci această putere se transformă în provocare. Personal nu cred că există vreo justificare pentru distrugerea de către *unii* a valorilor realizate de *alții*. Putem face altfel, putem nega valorile create de alții, afirmând că pentru noi nu sunt valori, dar distrugerea, inevitabil, înseamnă un pas înapoi. Un pas înapoi pentru noi toți.

Scriu acest editorial amintindu-mi întristată de 11 septembrie 2001, dar nu sunt mai veselă nici în acest prezent, care nu promite aproape nimic pentru viitorul apropiat care ne interesează. Pentru că vom merge la școală peste câteva zile, nu peste sute sau mii de zile. Pentru că acum, mult prea mulți sunt descumpăniți și privesc fără entuziasm spre ceea ce îi așteaptă în acest an școlar și nu se știe în câți dintre anii care vor veni...

Profesori, reprezentanții unor ministere ale educației din diverse țări, conducători ai delegațiilor echipelor participante la olimpiade internaționale m-au întrebat de mai multe ori despre secretele succesului concurenților români la olimpiadele de informatică. Ascultau aproape cu invidie că în *România* există un învățământ de informatică special, unde copiii doritori pot învăța tehnologia informației, algoritmica și programarea în opt ore pe săptămână. Că profesorii lucrează cu jumătate din clasă în laboratoare unde în cel mai rău caz stau 2-3 copii în fața unui calculator. Că pregătirea celor pasionați se completează la cercuri, la pregătirile speciale organizate la nivel local, județean sau național. Unii chiar nu credeau că aceste activități sunt realizate pe bază de voluntariat de către acei profesori care sunt fericiți să se ocupe de elevii doritori.

Mi-am adus aminte de bucuria cu care relatam toate cele înșirate mai sus... Acum privesc neîncrezătoare spre viitor: o clasă întreagă într-un laborator care a fost creat pentru 12-15 elevi..., patru ani de liceu într-o clasă care este de "mate-info", dar informatica este disciplină la alegere în cadrul examenului de *Bacalaureat*. De ce nu se schimbă în aceste condiții denumirea claselor de mate-info în mate-fizică-chimie-biologie-info? Probabil ar fi o denumire prea lungă...

Aș polemiza chiar și cu cei care susțin importanța formării unei culturi generale. Aș fi de acord cu ea dacă această "cultură generală" nu ar crește considerabil în fiecare an. Polemizez cu cei care compară cifrele statisticilor naționale privind promovabilitatea sau reușita la examene, cu cele din alte țări. Nu cred că nu ați auzit de câte un elev submediocru, plecat într-o școală din *Germania* sau *Franța* care, dintr-o dată, se trezește că este cel mai bun în clasa lui, de multe ori fără să învețe ceva. Ceva, undeva, nu se potrivește...

Actualul guvern, probabil se poate lăuda cu multe realizări. Nu sunt analist politic și nici nu vreau să încerc să vorbesc despre alte domenii. Dar, în învățământ sigur șchioapătă ceva. Totul începe cu lipsa revoltătoare a respectului față de munca prestată de profesori (vezi salarii, premii, mediatizare etc.) și terminând cu faptul că nimănui nu-i trece prin cap să se consulte cu ei, să țină cont măcar de părerea acelora care au avut și mai au disponibilitatea să ajute.

Chiar cred că, începând din antichitate, profesorul nu a fost nicicând atât de privat de respect ca acum. Nu este respectat de guvernanți, de oficialități, de elevi și de părinții lor care cred că dacă pot cheltui în trei zile salariul pe o lună a unui profesor, li se justifică pretenția de superioritate. Cred că nota de dramatism din ceea ce mi s-a întâmplat ieri, spune mult. M-a oprit pe stradă un pușți să-mi vândă ceva. Am încercat să scap de el, dar a insistat. Până la urmă i-am zis să mă lase în pace că nu am bani. "***Sunt profesoară!***" am adăugat cu scopul de a-mi întări refuzul. Copilul s-a uitat oarecum cu compasiune la mine, și-a strâns lucrurile și a plecat fără să mai spună nimic.

Clara Ionescu

flash

WinRar 3.0 (www.rarlab.com)

Pe data de 15 mai 2002 a fost lansată o nouă versiune a popularului program de arhivare WinRar. Una dintre cele mai importante îmbunătățiri este posibilitatea folosirii unui dicționar cu dimensiunea de 4 MB. Versiunea WinRar 3.0 este distribuită în regim shareware; pot fi descărcate gratuit variante localizate în diferite limbi, inclusiv româna.

Borland JBuilder 7

Programatorii în Java au la dispoziție, începând cu data de 20 mai, cea de-a șaptea versiune a mediului de programare JBuilder. Produsul este distribuit în patru variante și anume Enterprise, Standard Edition, Personal și MobilSet.

JDK 1.4 (www.sun.com)

Începând cu data de 29 mai este disponibilă pentru download o nouă versiune a pachetului JDK, destinat programatorilor care utilizează limbajul Java.

Mozilla 1.0 (www.mozilla.org)

După patru ani de dezvoltare, programul de navigare Mozilla a ajuns la versiunea finală pe data de 5 iunie. Acest browser folosește, la fel ca și Netscape Navigator 6, motorul Gecko și este distribuit gratuit.

WD Caviar 200 GB

Pe data de 24 iunie a fost lansat pe piață primul disc hard care are o capacitate de 200 GB și o viteză de 7200 rpm. Astfel, compania Western Digital este prima care atinge această barieră psihologică.

Microsoft Visual J# .NET

Pachetul Microsoft Visual Studio .NET are, începând cu data de 1 iulie, un nou "membru". Potrivit producătorilor, mediul Visual J# .NET a fost creat pentru a permite programatorilor în Java să beneficieze de avantajele oferite de limbajul XML.

nVidia nForce 2

Compania nVidia a lansat pe data de 16 iulie 2002 un nou chipset pentru plăci de bază compatibile cu procesoarele AMD. Cea mai importantă caracteristică a noului produs este placa video integrată care oferă performanțe asemănătoare cu o placă GeForce 4 MX.

QuickTime 6 (www.apple.com)

Apple a anunțat pe data de 15 iulie disponibilitatea versiunii cu numărul 6 a pachetului QuickTime. Acesta a apărut în versiuni pentru sistemele de operare Windows și MacOS. QuickTime 6 oferă suport pentru MPEG-4, 50 de formate de intrare/ieșire, numeroase codec-uri etc. Versiunea standard este distribuită gratuit, în timp ce QuickTime 6 Pro poate fi achiziționat la prețul de 29,99$.

Războiul procesoarelor

În această vară, competiția de pe piața procesoarelor dintre *Intel* și *AMD* a continuat, ambele companii lansând o mulțime de produse noi, destinate diferitelor tipuri de utilizatori.

Unul dintre cele mai importante evenimente care au avut loc în această vară a fost lansarea procesoarelor *Intel Pentium* 4 cu magistrala de 533 MHz. Evenimentul a avut loc pe data de 6 mai, când au devenit disponibile procesoare cu frecvențe de 2,26 GHz, 2,4 GHz și 2,53 GHz. Pe data de 26 august au fost lansate alte două modele care funcționau la frecvențe de 2,66 și 2,8 GHz. În aceeași zi au fost lansate alte două procesoare care folosesc o magistrală de 400 MHz și funcționează la frecvențe de 2,5 și 2,6 GHz.

Pe data de 15 mai a apărut primul procesor *Celeron* care folosește arhitectura *NetBurst*, la fel ca și *Pentium* 4. Noul procesor funcționează la frecvența de 1,7 GHz și are magistrala de 400 MHz. În aceeași zi a fost lansat și ultimul procesor *Intel* construit pe baza nucleului *Tualatin*. Acesta este un *Celeron* cu frecvența de 1,4 GHz și magistrala de 100 MHz. Chiar dacă are o frecvență cu 300 MHz mai mică, un procesor *Celeron* 1,4 GHz este mai performant decât un *Celeron* 1,7 GHz, fapt confirmat și de faptul că prețul său oficial (89$) era mai mare decât cel al procesorului care folosește noua arhitectură (83$). La mai puțin de o lună, pe data de 12 iunie, a fost lansat un nou model care funcționează la frecvența de 1,8 GHz și are prețul de 103$.

Intel a atins bariera de 2 GHz și pentru procesoarele destinate calculatoarelor portabile. Pe data de 24 iunie au fost lansate procesoare *Mobile Pentium* 4 cu frecvențe de 1,9 GHz și 2 GHz. În aceeași zi au apărut trei noi modele ale procesorului *Mobile Celeron* cu frecvențe de 1,33 GHz, 1,4 GHz și 1,5 GHz.

Un alt eveniment important a avut loc pe data de 8 iulie când *Intel* a anunțat disponibilitatea celei de-a doua generații de procesoare care folosesc o arhitectură de 64 de biți. Procesoarele poartă numele *Itanium* 2, au frecvențe de 900 MHz și 1 GHz, au o memorie *cache level* 3 de 1,5 MB sau 3 MB și prețuri cuprinse între 1338$ și 4226$.

Compania rivală *AMD* a lansat și ea pe piață numeroase modele noi în cursul acestei veri. Primul dintre acestea este *Athlon XP* 2200+ (1,8 GHz), primul procesor care folosește nucleul *Thoroughbred* bazat pe tehnologia de 0,13 µm. Familia a fost îmbogățită pe data de 21 august când au apărut procesoarele *Athlon XP* 2400+ (2 GHz) și *Athlon XP* 2600+ (2,13 GHz). Acest moment a marcat atingerea barierei de 2 GHz de către *AMD*, la aproape un an după *Intel*.

Un nou procesor *Athlon MP* a fost lansat pe data de 19 iunie. Acesta are frecvența de 1,73 GHz, denumirea sa este *Athlon MP* 2100+ și folosește vechea tehnologie de 0,18 µm. Primul procesor din această familie care folosește noua tehnologie de 0,13 µm, *Athlon MP* 2200+, a fost lansat pe data de 27 august; frecvența este de 1,8 GHz.

Cea de-a treia gamă de procesoare *AMD* (cele destinate calculatoarelor portabile) s-a îmbogățit și ea cu un nou membru pe data de 15 iulie când a fost lansat modelul *Mobile Athlon XP* 1800+, care funcționează la frecvența de 1,53 GHz.

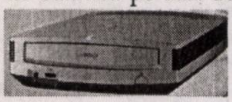
Chiar dacă nu este unul dintre principalii "jucători" pe piața procesoarelor, compania *VIA* a lansat și ea un nou produs pe data de 3 iunie. Este vorba despre procesorul *VIA C3* cu frecvența de 1 GHz. Astfel, la peste doi ani după *AMD* și *Intel*, compania *VIA* intră și ea în "era gigahertzilor". Noul procesor folosește o tehnologie de 0,13 mm, dispune de o memorie *cache level* 1 de 128 KB, și o memorie *cache level* 2 de 64 KB. Este folosit vechiul *Socket* 370, există variante cu magistrala de 100 MHz și 133 MHz și este implementat suport pentru seturile de instrucțiuni *MMX* și *3DNow*!

Noi unități CD-RW

În anii trecuți vitezele de citire ale unităților *CD-ROM* au crescut foarte repede de la 8× la peste 50×. În ultima perioadă un fenomen asemănător a avut loc pe piața unităților de inscripționare și reinscripționare a *CD*-urilor. În urmă cu doi ani, inscripționarea unui *CD* în zece minute (la viteza 8× - 1,2 MB/s) era considerată o performanță semnificativă. Anul trecut vitezele de inscripționare au crescut foarte mult, ajungându-se foarte repede la viteze de 32×, în timp ce reinscripționarea putea fi realizată doar cu viteze de cel mult 12×. În cursul acestei veri tendința de creștere a vitezelor s-a păstrat, fiind lansate o mulțime de produse noi cu viteze de inscripționare și reinscripționare din ce în ce mai mari.

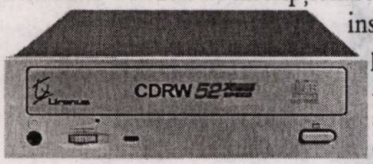
Primul record al verii a fost stabilit pe data de 8 mai 2002, când compania *ASUS* a anunțat disponibilitatea unității *ASUS CRW-4816A*. Din denumire rezultă că viteza de inscripționare este de 48×, iar cea de reinscripționare de 16×. Unitatea este capabilă să citească *CD*-uri cu viteza de 48×, aceeași cu viteza de inscripționare.

Un nou record în ceea ce privește viteza de reinscripționare a fost stabilit de *Yamaha* pe data de 3 iunie. Unitatea *Yamaha CRW-F1* era capabilă să reinscripționeze *CD*-uri la viteza de 24×. Vitezele de citire și reinscripționare erau de 44×, inferioare recordului stabilit de *ASUS* cu mai puțin de o lună în urmă.



Pe data de 22 iulie, compania taiwaneză *Ultima Technologie* a depășit recordul vitezei de inscripționare și l-a egalat pe cel al vitezei de reinscripționare. Noua unitate, comercializată sub denumirea *Artec*, era capabilă să inscripționeze *CD*-uri cu viteza de 52×, să reinscripționeze la viteza de 24× și să citească la viteza de 52×.

Poziția de lider a companiei taiwaneze a fost păstrată doar câteva zile, pe data de 25 iulie fiind anunțat un nou produs din gama *Uranus* a companiei *Traxdata*. Vitezele de inscripționare și citire sunt aceleași (52×), dar cea de reinscripționare este de 32×. Astfel, *Traxdata* produce cea mai performantă unitate *CD-RW* din punct de vedere al vitezei de inscripționare, reinscripționare și citire a *CD*-urilor.



Timpul liber

Trei jocuri au atras în mod deosebit atenția în această vară. Primul dintre ele este *Neverwinter Nights*, un *RPG* produs de *Bioware*, compania care a creat celebra serie *Baldur's Gate*. Cea mai importantă inovație este faptul că este folosit noul sistem de reguli *AD&D* 3. În rest, conform așteptărilor, cei de la *Bioware* au reușit să creeze o lume fantastică atrăgătoare, pe care pasionații de jocuri o vor descoperi pe parcursul multor săptămâni în care le va fi din ce în ce mai greu să plece din fața ecranului.

Unul dintre cele mai așteptate evenimente din industria jocurilor a avut loc pe data de 3 iulie când *Blizzard* a anunțat că *Warcraft* III - *Reign of Chaos* este disponibil pe piață. Celebra serie de *RTS*-uri beneficiază acum de o grafică *3D* impresionantă, care folosește tehnologii de ultimă oră. Așa cum spuneam în prezentarea din numărul 11/2 (februarie 2001) pe lângă rasele cunoscute din *Warcraft* I și II (*orc* și *human*), au apărut două rase noi și anume *undead* și *night elves*.

Iubitorii șahului se pot bucura, începând din 27 august, de noul *ChessMaster* 9000. Cea mai bine vândută serie de jocuri de șah este acum disponibilă pentru *PC*-uri și consolele *GameBoy Advance*.

flash

Radeon 9000 și Radeon 9700

Începând cu 18 iulie, familia chip-urilor oferite de ATI pentru plăci video are noi membri. Radeon 9000 și Radeon 9000 Pro oferă suport complet pentru DirectX 8.1 și folosesc 64 MB de memorie DDR. Cel mai performant nou membru al familiei este Radeon 9700; lansarea acestui produs a permis companiei ATI să devină lider mondial în ceea ce privește performanțele chip-urilor pentru plăci grafice. Radeon 9700 depășește performanțele unui GeForce 4 Ti 4600, produsul de vârf al companiei rivale nVidia. Noul chip oferă suport complet pentru specificațiile AGP 8x și folosește 128 MB de memorie DDR. Varianta destinată profesioniștilor nu se numește Radeon 9700 Pro, cum era de așteptat, ci Fire GL X1. Se pare că ATI intenționează să sublinieze diferențele dintre cipurile destinate utilizatorilor obișnuiți și cele pentru profesioniști. Principala diferență dintre Radeon 9700 și Fire GL X1 este dată de faptul că cea de-a doua folosește 256 MB de memorie DDR.

CorelDraw Graphics Suite 11

Compania Corel a lansat pe data de 1 august 2002 a nouă versiune a pachetului de programe pentru prelucrare grafică. CorelDraw Graphics Suite 11 conține aplicațiile CorelDraw 11, Corel Photo-Paint 11 și Corel R.A.V.E 2 și este disponibil în variante pentru sistemele de operare Windows și MacOS.

Borland Delphi 7 Studio

Popularul mediu de programare Delphi, oferit de compania Borland Software, a ajuns pe data de 6 august la cea de-a șaptea versiune. Delphi 7 Studio este distribuit în patru variante și anume Architect, Enterprise, Professional și Personal.

WinAmp 3 (www.winamp.com)

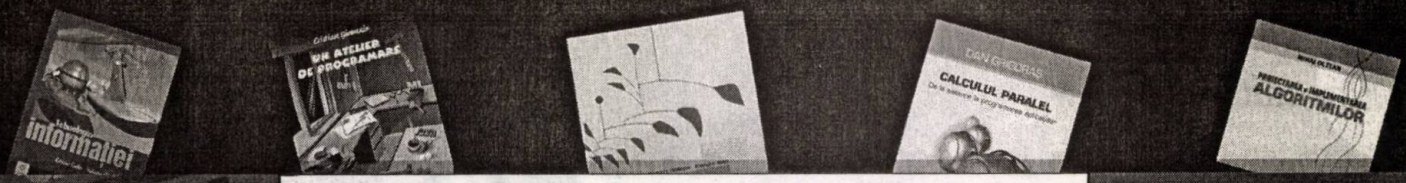
După mai mult de doi ani de dezvoltare și o lungă perioadă de testare BETA, compania NullSoft a anunțat disponibilitatea produsului WinAmp 3. Acesta poate fi descărcat gratuit și oferă o mulțime de facilități noi față de versiunile WinAmp 2.x.

Netscape 7 (www.netscape.com)

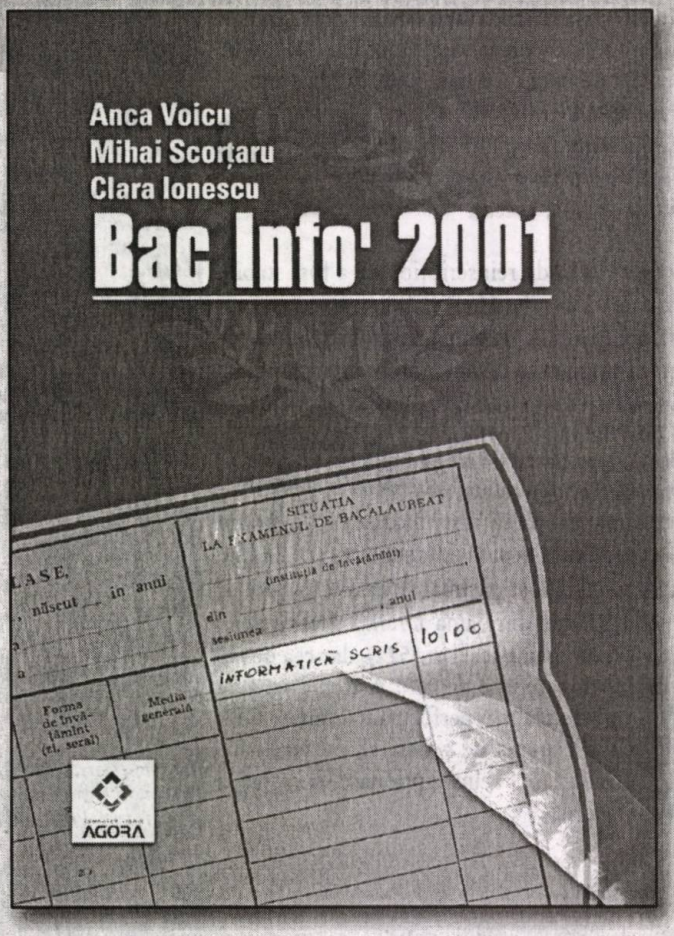
Primul program de navigare care a ajuns la versiunea 7 este Netscape Navigator. Netscape 7 a fost lansat pe data de 28 august, folosește o nouă versiune îmbunătățită a motorului Gecko și este cel mai rapid browser Netscape creat până acum.

MacOS X 10.2

Cunoscută sub numele de cod Jaguar, versiunea 10.2 a sistemului de operare MacOS X oferit de Apple a fost lansată pe data de 23 august. Potrivit producătorilor, MacOS X 10.2 combină siguranța oferită de UNIX cu ușurința în utilizare specifică MacIntosh.



COMPUTER LIBRIS AGORA



Bac Info' 2001 conține 20 de variante de subiecte pentru Bacalaureat. Fiecare variantă conține câte patru subiecte. Primul dintre ele conține trei subpuncte care testează capacitățile candidaților de a recunoaște valoarea de adevăr a unei propoziții, de a determina efectul unui anumit algoritm și de a găsi o anumită greșeală într-un algoritm. Al doilea subiect conține doua subpuncte; primul testează deprinderea candidaților de a folosi limbajul pseudocod, iar al doilea cea de a folosi un limbaj de programare (Pascal sauC++). Al treilea subiect este o problemă pentru rezolvarea căreia este necesară folosirea unor structuri dinamice de date, iar al patrulea este o problemă pentru a cărei rezolvare trebuie folosită metoda backtracking.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

Belgrad, Iugoslavia

BOI 2002

Primul concurs internațional important al anului la care au participat elevi români a fost Balcaniada. Vă prezentăm acum enunțurile problemelor propuse spre rezolvare la această a zecea ediție a competiției.

P060201: Robot

În viitorul apropiat, roboții ar trebui să transporte gustările pe care le vor servi participanții la *Olimpiada Balcanică de Informatică* în timpul concursului, folosind o singură tavă de formă pătratică. Din nefericire, este foarte probabil ca drumul dintre bucătărie și sala de concurs să fie "presărat" cu tot felul de obstacole, deci un robot nu va putea să transporte tăvi oricât de mari. Sarcina voastră este să determinați care este dimensiunea maximă a unei tăvi care poate fi transportată.

Drumul pe care ar trebui să îl parcurgă robotul este printr-un coridor cu pereții paraleli și care nu poate avea decât cotituri cu 90°. Coridorul începe în direcția axei *Ox* pozitive. Obstacolele constau în stâlpi cu dimensiuni neglijabile și sunt reprezentate prin puncte. Pentru ca robotul să poată ajunge în sala de concurs, tava pe care o poartă nu poate să se lovească de pereți sau de stâlpi, dar poate să îi atingă cu marginile. Robotul și tava sa se pot deplasa doar paralel cu axele de coordonate. Dimensiunile robotului sunt întotdeauna mai mici decât cele ale tăvii, iar robotul este în întregime acoperit de tava pe care o poartă.

Date de intrare

Prima linie a fișierului **ZAD1.DAT** conține un număr întreg m ($1 \le m \le 30$) care reprezintă numărul segmentelor liniare de pereți. Următoarele $m + 1$ linii ale fișierului de intrare conțin coordonatele x și y ale tuturor punctelor de cotitură (inclusiv ale punctelor terminale) ale peretelui care inițial, are coordonata y mai mare. Similar, următoarele $m + 1$ linii conțin coordonatele x și y ale tuturor punctelor de cotitură (inclusiv ale punctelor terminale) ale peretelui care inițial, are coordonata y mai mică. Următoarea linie conține un număr întreg n ($0 \le n \le 100$) care reprezintă numărul obstacolelor. Următoarele n linii conțin coordonatele x și y ale obstacolelor. Toate coordonatele sunt numere întregi a căror valoare absolută este mai mică decât 32001.

Date de ieșire

Fișierul de ieșire **ZAD1.RES** va conține o singură linie pe care se va afla un întreg care va reprezenta lungimea maximă a laturii tăvii.

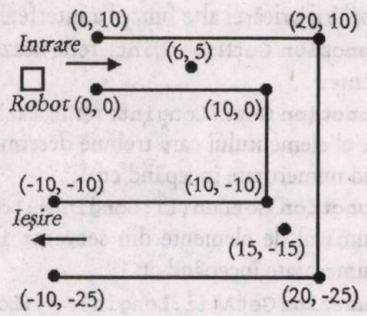
Exemplu

ZAD1.DAT

```
3
0 10
20 10
20 -25
-10 -25
0 0
10 0
10 -10
-10 -10
2
15 -15
6 5
```

ZAD1.RES

```
5
```



P060202: Secvențe

Se consideră n secvențe ($2 \le n \le 10$), fiecare fiind formată din numere întregi cuprinse în intervalul $[0, 2^{31} - 1]$ dispuse în ordine crescătoare. Toate elementele din toate secvențele sunt diferite și, în total, nu sunt mai mult de 50000 elemente.

Elementele din secvențe nu sunt cunoscute, dar există posibilitatea apelării unei funcții pentru a obține valoarea unui element aflat într-o poziție a unei secvențe.

Se cere să se determine, utilizând un număr minim de apeluri ale funcției care furnizează valoarea unui element, cel de-al k-lea element din secvența rezultată în urma interclasării celor n secvențe inițiale.

Cel de-al k-lea element va fi scris pe prima linie a fișierului **ZAD2.RES**.

Instrucțiuni pentru programatorii în C

În codul sursă trebuie inserată linia:

```
#include "mergem.h"
```

Fișierul sursă trebuie linkeditat împreună cu fișierul **mergem.o**.

Interfața C vă pune la dispoziție următoarele funcții:

- **void** `Init()`: este folosită pentru inițializare și va fi apelată, obligatoriu, la începutul programului, înaintea apelării oricărei alte funcții a interfeței;
- **long** `GetN()`: returnează numărul de secvențe;

- **long** `GetK()`: returnează numărul de ordine al elementului care trebuie determinat, elementele fiind numerotate începând cu 1;
- **long** `GetLen(long ii)`: returnează numărul de elemente din secvența `ii`, secvențele fiind numerotate începând cu 1;
- **long** `GetA(long ii,long jj)`: returnează elementul cu numărul de ordine `jj` din secvența `ii`, elementele dintr-o secvență fiind numerotate începând cu 1;
- **void** `Finish()`: funcția va fi apelată obligatoriu înaintea terminării execuției programului.

Instrucțiuni pentru programatorii în Pascal

În codul sursă trebuie inserată linia:

```
uses mergem;
```

Interfața *Pascal* vă pune la dispoziție următoarele funcții și proceduri:

- **procedure** `init`: este folosită pentru inițializare și va fi apelată, obligatoriu, la începutul programului, înaintea apelării oricărei alte funcții a interfeței;
- **function** `GetN:Longint`: returnează numărul de secvențe;
- **function** `GetK:Longint`: returnează numărul de ordine al elementului care trebuie determinat, elementele fiind numerotate începând cu 1;
- **function** `GetLen(ii:Longint):Longint`: returnează numărul de elemente din secvența `ii`, secvențele fiind numerotate începând cu 1;
- **function** `GetA(ii:Longint; jj:Longint):Longint`: returnează elementul cu numărul de ordine `jj` din secvența `ii`, elementele dintr-o secvență fiind numerotate începând cu 1;
- **procedure** `Finish`: procedura va fi apelată obligatoriu înaintea terminării execuției programului.

Exemplu

Comunicare cu modulul	
apel	*valoare returnată*
`GetN`	3
`GetK`	3
`GetLen(1)`	3
`GetLen(2)`	4
`GetLen(3)`	4
`GetA(1,1)`	10
`GetA(2,1)`	35
`GetA(3,1)`	210
`GetA(1,2)`	40
`GetA(2,2)`	150

ZAD2.OUT

```
40
```

P060203: Tur

Agenția de turism *Programatorii* trebuie să organizeze turul mănăstirilor din munții *Fruška Gora*. În acești munți există n ($3 \le n \le 1000$) mănăstiri, identificate prin numere întregi cuprinse între 1 și n. Din cauza timpului scurt nu se pot vizita toate mănăstirile, ci doar k ($3 \le k \le n$) dintre ele. Există drum direct între fiecare două mănăstiri din *Fruška Gora*, dar cum acestea sunt drumuri de munte, sunt foarte înguste, și au un singur sens de mers.

Trebuie să scrieți un program care să determine un tur care să treacă pe la exact k mănăstiri, turul trebuie să înceapă și să se termine la aceeași mănăstire, iar celelalte mănăstiri din tur să nu fie vizitate de două ori.

Date de intrare

Pe prima linie a fișierului **ZAD3.DAT** se află numerele întregi n și k, separate printr-un spațiu. Fiecare dintre următoarele $n \cdot (n - 1) / 2$ linii conține două numere întregi a și b ($1 \le a, b \le n$) care indică faptul că există drum de la mănăstirea a la mănăstirea b.

Date de ieșire

În cazul în care există un tur care respectă condițiile date, în fișierul **ZAD3.RES** se vor scrie k întregi care reprezintă mănăstirile din tur în ordinea vizitării lor. Dacă există mai multe astfel de tururi, se va scrie unul din ele. Dacă nu există soluție, în fișier se va scrie doar valoarea 0.

Exemplu

ZAD3.DAT

```
6 3
3 4
1 3
3 5
3 6
3 2
4 1
4 5
4 6
4 2
1 5
1 6
1 2
5 6
2 5
6 2
```

ZAD3.RES

```
1 3 4
```

P060204: Extratereștrii

Extratereștrii ne vizitează planeta cu intenția evidentă de a găsi noi specii pentru grădina lor zoologică. După intrarea pe orbita Pământului ei s-au poziționat deasupra orașului *Belgrad*, unde au detectat o anumită formă de activitate. La apropierea de suprafața Pământului ei au văzut un grup de ființe semiinteligente. Aceste creaturi sunt actualii concurenți de la *BOI* 2002 care participă la excursia de după intensul concurs. Extratereștrii doresc să captureze toți cei n ($2 \le n \le 100000$) concurenți.

Extratereștrii folosesc o undă tractoare pentru a-și captura prada. Cu ajutorul undei tractoare pot fi capturate toate ființele aflate în interiorul sau pe circumferința unei suprafețe circulare. Cu cât raza undei tractoare crește (raza cercului la sol), cu atât consumul de energie este mai mare. Fiind extrem de inteligenți, extratereștrii sunt mai avansați

în domeniul științelor sociale decât în domeniul programării. Din această cauză, ei vă cer ajutorul pentru a determina poziția navei lor spațiale astfel încât energia necesară pentru capturarea celor n concurenți să fie minimă.

Ajutați-i pe frații extratereștri! Scrieți un program care să găsească rază minimă a zonei circulare care cuprinde toți cei n concurenți și poziția optimă a navei spațiale (centrul zonei circulare de la sol).

Date de intrare

Pe prima linie a fișierului **ZAD4.DAT** se află un număr întreg n care reprezintă numărul participanților la *BOI 2002*. Fiecare din următoarele n linii conține două numere reale x_i și y_i ($-10000.0 \le x_i, y_i \le 10000.0$), reprezentând coordonatele celui de-al i-lea concurent.

Date de ieșire

Fișierul de ieșire **ZAD4.RES** va conține două linii. Pe prima se află un singur număr real care reprezintă raza undei tractoare. A doua linie conține două numere reale, x și y, reprezentând coordonatele navei spațiale. Numerele din fișierul de ieșire vor fi scrise folosind două zecimale exacte.

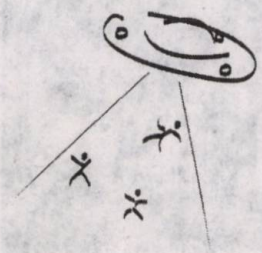
Exemplu

ZAD4.DAT

```
6
8.0 9.0
4.0 7.5
1.0 2.0
5.1 8.7
9.0 2.0
4.5 1.0
```

ZAD4.RES

```
5.00
5.00 5.00
```



P060205: Arbore

O enumerare a nodurilor unui arbore este orice secvență ale cărei elemente sunt nodurile unui arbore iar fiecare nod apare exact o dată. Dacă oricare două elemente consecutive dintr-o secvență sunt noduri pentru care distanța în arbore nu este mai mare decât d, atunci respectiva enumerare este denumită d-enumerare.

Se poate demonstra că pentru orice arbore există cel puțin o 3-enumerare. Scrieți un program care să determine o 3-enumerare pentru un arbore dat.

Date de intrare

Pe prima linie a fișierului text **ZAD5.DAT** se află numărul n ($1 \le n \le 30000$), reprezentând numărul de noduri ale arborelui. Nodurile sunt numerotate de la 1 la n. Pe fiecare dintre următoarele n - 1 linii se află câte două numere întregi care reprezintă extremitățile uneia dintre muchiile arborelui.

Date de ieșire

Fișierul text **ZAD5.RES** va conține o 3-enumerare a arborelui dat; fiecare linie a fișierului va conține câte un element al acestei 3-enumerări.

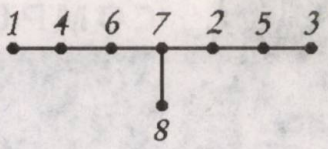
Exemplu

ZAD5.DAT

```
8
1 4
4 6
6 7
3 5
7 2
2 5
8 7
```

ZAD5.RES

```
3
2
6
1
4
8
7
5
```



P060206: Hartă veche

În biblioteca unei mănăstiri a fost găsită o carte veche cu hărți. Una dintre acestea conține doar castele și drumuri. Drumurile leagă castelele și nu se intersectează. Se poate călători între oricare două castele utilizând aceste drumuri. Pe hartă apar proprietăți care sunt delimitate de drumuri. Fiecare proprietate este delimitată de o linie poligonală simplă închisă formată din drumuri.

Taxa pentru o proprietate este proporțională cu numărul de castele aflate în interiorul proprietății. Cea mai mică taxă este pentru acea proprietate care nu are nici un castel sau drum în interiorul său.

Trebuie să scrieți un program care determină numărul de proprietăți de pe o hartă dată, care sunt delimitate de exact k drumuri și pentru care taxa este minimă.

Date de intrare

Pe prima linie a fișierului text **ZAD6.DAT** se află un număr întreg n ($3 \le n \le 1000$), reprezentând numărul de castele (acestea sunt identificate prin numere cuprinse între 1 și n). Pe fiecare dintre următoarele n linii se găsesc coordonatele întregi x și y ($0 \le x, y \le 10000$) ale unui castel, numărul d al castelelor vecine (cele cu care este legat direct) și cele d numere de ordine ale castelelor vecine. Pe ultima linie a fișierului se află numărul întreg k ($3 \le k \le 1000$).

Date de ieșire

Fișierul de ieșire **ZAD6.RES** va conține o singură linie pe care se va afla un număr întreg, reprezentând numărul de proprietăți pentru care taxa este minimă și care sunt delimitate de k drumuri.

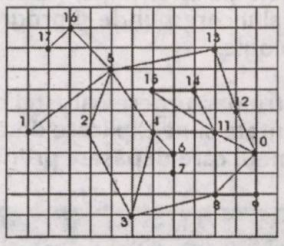
Exemplu

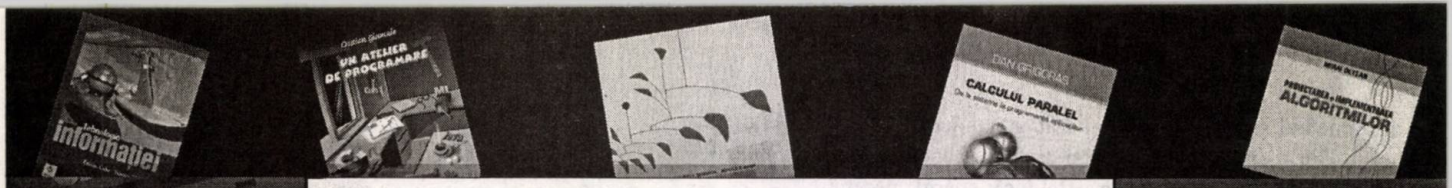
ZAD6.DAT

```
17
1 5 2 2 5
4 5 3 1 3 5
6 1 3 2 4 8
7 5 4 3 5 6 11
5 8 5 13 16 1 2 4
8 4 2 4 7
8 3 1 6
10 2 4 9 10 11 3
12 2 2 10 8
12 4 4 12 11 8 9
10 5 6 14 15 4 8 10 13
11 6 2 10 13
10 9 3 5 11 12
9 7 2 15 11
7 7 2 14 11
3 10 2 5 17
2 9 1 16
4
```

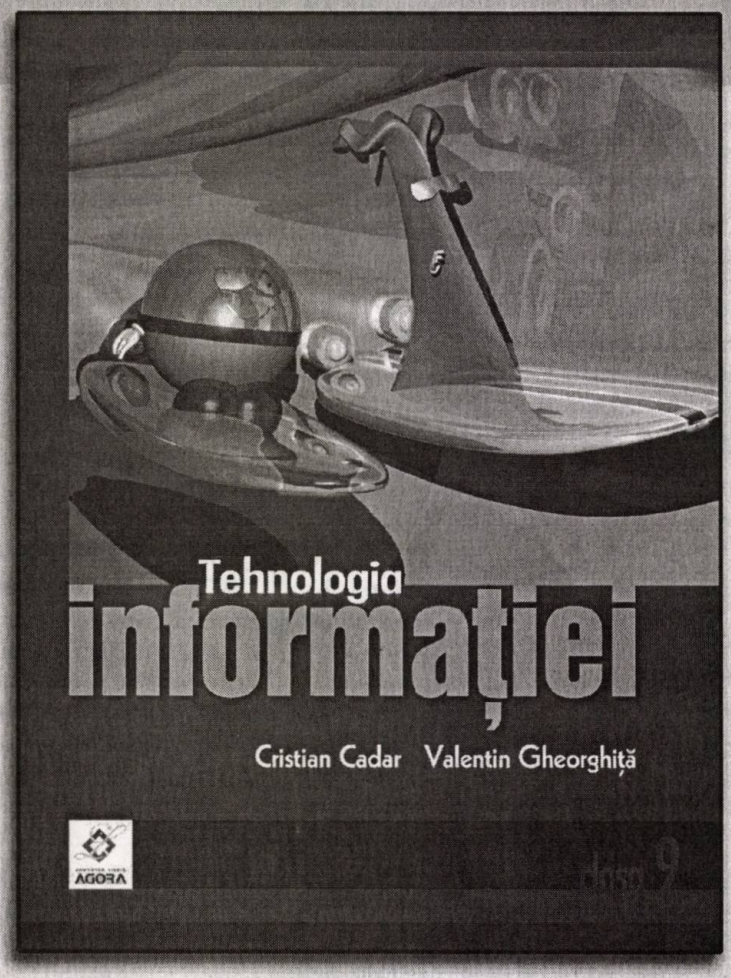
ZAD6.RES

```
2
```





COMPUTER LIBRIS AGORA



Lucrarea de față își propune să pregătească cititorul în scopul utilizării unui calculator personal. Ea urmărește programa școlară, aprobată de Ministerul Educației Naționale, pentru disciplina "Tehnologia informației", disciplină în planul de învățământ a claselor a IX-a la profilul teoretic și pedagogic. Manualul corespunde și programei de la disciplina "Informatică - Tehnologii asistate pe calculator", profilul tehnologic.

Pentru parcurgerea acestei cărți nu sunt necesare cunoștințe prealabile în domeniu, termenii și noțiunile prezentate sunt explicate detaliat în carte. Exemplele sunt sugestive, cartea nu este încărcată inutil cu informații care depășesc programa amintită.

COMPUTER LIBRIS AGORA

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

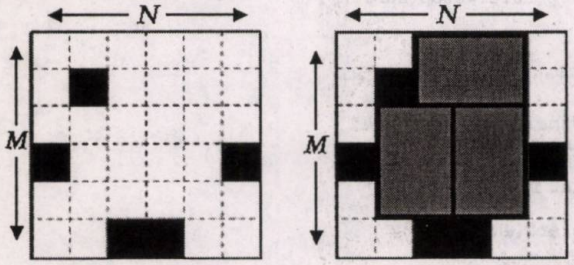
Košice, Slovacia

CEOI 2002

Cea de-a noua ediție a Olimpiadei de Informatică a Europei Centrale s-a desfășurat în perioada 30 iunie - 6 iulie 2002 la Košice, în Slovacia. În continuare veți găsi enunțurile celor șase probleme propuse spre rezolvare la această competiție.

P060207: Bugs Integrated, Inc.

Bugs Integrated, Inc. este o întreprindere specializată în producerea *chip*-urilor de memorie. Firma se pregătește să înceapă producția unor noi *chip*-uri (*Q-RAM*) cu capacitatea de șase teraocteți. Fiecare *chip* este format din șase pătrățele, având latura egală cu unitatea, aranjate în dreptunghiuri de dimensiuni 2 × 3. *Chip*-urile *Q-RAM* sunt fabricate folosind o placă de silicon de formă dreptunghiulară împărțită în $N \times M$ pătrățele, având latura egală cu unitatea. Toate pătrățelele sunt testate cu atenție și cele defecte sunt marcate cu negru.



Urmează ca din placa de silicon să se taie *chip*-urile. Fiecare *chip* este format din 2 × 3 (sau 3 × 2) pătrățele cu latura egală cu unitatea. Bineînțeles, *chip*-urile nu pot conține pătrățele defecte. Este posibil ca prin tăierea plăcii să nu se folosească toate pătrățelele nemarcate. Firma ar dori să piardă cât mai puține pătrățele nemarcate. În concluzie, ei doresc să afle modalitatea de tăiere a plăcii, astfel încât să rezulte un număr maxim de *chip*-uri.

Se consideră dimensiunile a mai multor plăci de silicon și pentru fiecare dintre acestea lista tuturor pătrățelelor defecte. Scrieți un program care calculează pentru fiecare placă numărul maxim de *chip*-uri care pot fi tăiate din acestea.

Date de intrare

Prima linie a fișierului de intrare **BUGS.IN** conține un singur număr întreg D ($1 \le D \le 5$), reprezentând numărul plăcilor de silicon.

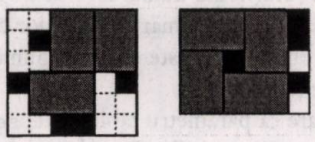
Urmează D blocuri de date, corespunzând celor D plăci de silicon. Prima linie a fiecărui bloc conține trei numere întregi N ($1 \le N \le 350$), M ($1 \le M \le 10$) și K ($0 \le K \le M \times N$) separate printr-un singur spațiu. N reprezintă dimensiunea pe orizontală, corespunzătoare coordonatei x (numărul coloanelor) în care s-a împărțit placa, M reprezintă dimensiunea pe verticală, corespunzătoare coordonatei y (numărul liniilor), iar K este numărul pătrățelelor defecte. Următoarele K linii conțin lista pătrățelelor defecte. Pe fiecare linie se află două numere întregi x și y ($1 \le x \le N$, $1 \le y \le M$) care reprezintă coordonatele a câte unui pătrățel defect (colțul din stânga-sus are coordonatele [1, 1], colțul din dreapta-jos are coordonatele [N, M]).

Date de ieșire

Corespunzător fiecărei plăci descrise în fișierul de intrare, se va scrie în fișierul de ieșire **BUGS.OUT** un singur număr, reprezentând numărul maxim de *chip*-uri care pot fi tăiate din placa respectivă.

Exemplu

BUGS.IN	BUGS.OUT
2	3
6 6 5	4
1 4	
4 6	
2 2	
3 6	
6 4	
6 5 4	
3 3	
6 1	
6 2	
6 4	



Timp maxim de execuție/test: 1 secundă
Memorie disponibilă: 16 MB

P060208: Batalionul Cuceritorului

În istoria omenirii există câteva bătălii ciudate, cum este și cea care a avut loc în *Franța*, în 1747. În satul *Bassignac-le-Haut*, situat pe malul stâng al râului *Dordogne*, deasupra

digului *Chastang* a existat o cetate. De la dig spre cetate erau amplasate trepte imense. Într-o bună zi s-a apropiat de cetate un batalion condus de *Cuceritor*.

Cuceritorul a fost întâmpinat în dreptul cetății de șeful gărzii de apărare. Acesta, având la dispoziție doar o mică parte din garda sa, i-a spus *Cuceritorului*: "*Văd că ai mulți soldați, așezați în spatele tău pe trepte. Îți propun următorul joc: la fiecare pas îți vei împărți soldații în mod arbitrar în două grupuri. Apoi, eu voi hotărî care dintre cele două grupuri va putea să rămână și care va trebui să plece acasă. Fiecare soldat care rămâne se va urca pe treapta următoare. Dacă cel puțin un soldat de-al tău ajunge pe treapta cea mai înaltă, te voi declara învingător, altfel vei fi învins. Și în acest caz te voi distruge...*"

Cuceritorului i-a plăcut ideea, a acceptat condițiile și a început "cucerirea". Vei avea rolul *Cuceritorului*; există N trepte până la cetate ($2 \le N \le 2000$) și ai cel mult 1000000000 de soldați. Corespunzător fiecărei trepte, se dă numărul de soldați care, în momentul începerii jocului, se află pe treapta respectivă. Treptele sunt numerotate de la 1 (corespunzând treptei cele mai înalte) la N care este treapta cea mai joasă. În momentul începerii jocului, pe treapta având numărul de ordine 1, nu va sta nici un soldat.

Pentru fiecare configurație inițială de joc dată, în cazul în care există strategie de câștig, (adică, poți câștiga indiferent de mutările adversarului), programul tău va trebui să câștige. În caz contrar, programul trebuie să joace corect și să piardă.

Aceasta este o problemă interactivă; vei juca împotriva unei biblioteci. La fiecare pas programul tău va specifica un grup de soldați. Biblioteca va returna 1 sau 2 specificând grupul de soldați care va rămâne (1 reprezintă grupul specificat de program, iar 2 reprezintă grupul format din restul soldaților). Dacă jocul se sfârșește (fie din cauza că ai câștigat, fie deoarece nu mai există soldați în joc), biblioteca va încheia execuția programului tău în mod corect. Execuția programului nu se poate încheia în nici un alt mod.

Interfața bibliotecii

Biblioteca `libconq` furnizează două subprograme:

- `start`: returnează numărul N al treptelor și încarcă tabloul `stairs` cu numărul soldaților care stau pe ele (valoarea `stairs[i]` este egală cu numărul soldaților care stau pe treapta i);
- `step`: are ca parametru tabloul `subset` (având cel puțin N elemente pentru *FreePascal* și cel puțin $N + 1$ elemente pentru *C/C++*), reprezentând grupul de soldați descris de tine, și returnează 1 sau 2 conform descrierii de mai sus; grupul de soldați este specificat de numărul soldaților aflați pe fiecare treaptă, la fel ca în cazul funcției `start` (valoarea `subset[i]` este egală cu numărul soldaților aparținând primului grup care se află pe treapta i).

Dacă greșești în precizarea unui grup de soldați, jocul se va termina și *Cuceritorul* va fi învins. Treptele sunt numerotate începând cu 1 atât pentru *FreePascal*, cât și pentru *C/C++*.

Declarațiile subprogramelor în FreePascal

```
procedure start(var N:Longint; var stairs:
                                  array of Longint);
function step(subset:array of Longint):
                                          Longint;
```

Declarațiile subprogramelor în C/C++

```
void start(int *N,int *stairs);
int step(int *subset);
```

Exemple de utilizare a bibliotecii

Urmează exemple de utilizare a bibliotecii în *FreePascal*, respectiv în *C/C++*; cele două fragmente de program au același efect: pornesc jocul și joacă o rundă, alegând aleator treptele pe care va sta fiecare soldat. Programul tău va trebui să cicleze la infinit.

Exemplu pentru FreePascal:

```
uses libconq;
var stairs:array[1..2000] of Longint;
    subset:array[1..2000] of Longint;
    i,N,result:Longint;
...
start(N,stairs);
...
for i:=1 to N do
  if random(2)=0 then
    subset[i]:=0
  else
    subset[i]:=stairs[i];
  result:=step(subset);
...
```

Exemplu pentru C/C++:

```
#include "libconq.h"
int stairs[2001];
int subset[2001];
int i,N,result;
...
start(&N,stairs);
...
for (i=1;i<=N;i++)
  if (rand()%2==0)
    subset[i]=0;
  else
    subset[i]=stairs[i];
result=step(subset);
...
```

În *Pascal* vei utiliza **`uses libconq;`** în *C/C++* vei folosi `#include "libconq.h"` și va trebui să comunici compilatorului și argumentul **`libconq.c`**.

Exemplu de comunicare cu biblioteca

Program	*Bibliotecă*
`start(N,stairs)`	$N = 8$, `stairs=(0,1,1,0,3,3,4,0)`
`step((0,1,0,0,1,0,1,0))`	*returnează* 2
`step((0,1,0,0,0,1,0,0))`	*returnează* 2
`step((0,0,0,3,2,0,0,0))`	*returnează* 1
`step((0,0,2,0,0,0,0,0))`	*returnează* 2
`step((0,1,0,0,0,0,0,0))`	*returnează* 2
`step((0,1,0,0,0,0,0,0))`	*nu returnează nimic; ai câștigat*

Timp maxim de execuție/test: 1 secundă
Memorie disponibilă: 16 MB

P060209: Un gărduleț decorativ

Richard tocmai a terminat construcția noii sale case. Singurul lucru care mai lipsește este un gărduleț decorativ de lemn. Dar, ca de obicei, habar nu are cum să-l conceapă, deci a decis să comande garduleţe de la o firmă. Întâmplător a găsit un catalog 2002 a firmei *ACME Fence*. După studierea acestuia și-a dat seama ce înseamnă de fapt un gărduleț decorativ.

Un asemenea gărduleț este format din N scânduri, așezate în linie, unele lângă altele, în poziție verticală. Gărduleţul va arăta bine doar dacă îndeplinește următoarele condiții:

- scândurile au lungimi diferite și anume 1, 2, ..., N unități;
- orice scândură care are doi vecini este fie mai lungă decât cei doi vecini ai săi, fie este mai scurtă decât aceștia (observați că astfel vârfurile scândurilor formează o linie frântă care fie urcă, fie coboară de la scândură la scândură);

Rezultă că fiecare gărduleț format din N scânduri poate fi descris în mod unic ca fiind o permutare a_1, ..., a_N a numerelor 1, ..., N, astfel încât:

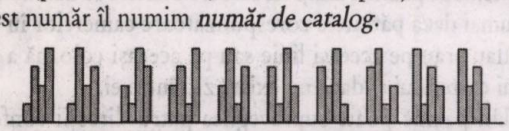
$$\forall i;\ 1 < i < N:\ (a_i - a_{i-1}) \cdot (a_i - a_{i+1}) > 0.$$

De asemenea, oricare astfel de permutare descrie un gărduleț.

Este evident că astfel, din N scânduri se pot forma multe gărdulețe diferite. Pentru a pune în catalog diferitele modele de gărdulețe într-o oarecare ordine, directorul de vânzări a firmei *ACME* a decis să le numeroteze în felul următor: gărdulețul A (reprezentat de permutarea a_1, ..., a_N) se află în catalog în fața gărdulețului B (reprezentat de permutarea b_1, ..., b_N) dacă și numai dacă există o valoare i, pentru care avem:

$$\forall j < i:\ a_j = b_j \text{ și } a_i < b_i.$$

De asemenea, pentru a decide care din cele două gărdulețe se va afla mai în față în catalog, se iau permutările corespunzătoare lor, se caută prima poziție în care ele diferă și se compară valorile aflate pe aceste poziții. Toate gărdulețele care se pot forma din N scânduri sunt numerotate (pornind de la 1) în ordinea apariției în catalog. Acest număr îl numim ***număr de catalog***.



Toate gărdulețele formate din N = 4 scânduri, ordonate pe baza numărului de catalog.

După ce *Richard* a studiat cu atenție toate gărdulețele, a decis să comande câteva ditre ele. Pentru fiecare model el și-a notat numărul de scânduri din care acesta este format, precum și numărul de catalog al acestuia. Într-o zi, a fost vizitat de prietenii săi și a vrut să le arate forma gărdulețelor comandate. Dar, zăpăcit cum era de obicei, el a rătăcit catalogul. Ajutați-l să deducă din datele pe care și le-a notat forma gărdulețelor comandate.

Date de intrare

Prima linie a fișierului de intrare **FENCE.IN** conține un număr întreg K ($1 \le K \le 100$), reprezentând numărul testelor conținute în fișier. Următoarele K linii conțin fiecare câte un set de date.

Pe fiecare din cele K linii se află două numere întregi N ($1 \le N \le 20$) și C, separate printr-un singur spațiu. N reprezintă numărul scândurilor din gărduleț, iar C este numărul de catalog al gărduleţului respectiv.

Se garantează că numărul gărdulețelor care se pot forma cu 20 de scânduri este un număr întreg care se poate reprezenta pe 64 de biți (`long long` în *C/C++*, `int64` în *FreePascal*). De asemenea, puteți fi siguri că fișierul de intrare este corect, mai exact valoarea cea mai mică a lui C este 1 și nu depășește numărul gărdulețelor care se pot forma din N scânduri.

Date de ieșire

Corespunzător fiecărui set de date din fișierul de intrare scrieți în fișierul de ieșire **FENCE.OUT** permutarea a_1, ..., a_N corespunzătoare gărduleţului format din N scânduri care are numărul de catalog C. Elementele permutării se vor scrie pe o singură linie și se vor despărți prin câte un spațiu.

Exemplu

FENCE.IN	FENCE.OUT
2	1 2
2 1	2 3 1
3 3	

Timp maxim de execuție/test: 1 secundă
Memorie disponibilă: 1 MB

P060210: Autostrada și cei șapte pitici

A fost odată ca niciodată o țară în care trăiau mai multe familii de pitici. În *Țara Piticilor* fiecare familie locuia în câte o casă. Piticii își vizitau des prietenii din alte familii. Deoarece în *Țara Piticilor* nu exista nici un fel de conflict, se întâmpla frecvent ca toți piticii să-i viziteze pe toți ceilalți într-o perioadă scurtă de timp.

O dată, oamenii care locuiau în țările vecine *Țării Piticilor* au hotărât să construiască unele autostrăzi în linii drepte. Deoarece oamenii nu aveau nici un fel de rețineri față de pitici, ei au proiectat anumite autostrăzi astfel încât acestea urmau să traverseze *Țara Piticilor*. Piticii au descoperit aceste planuri și au fost extrem de nefericiți, deoarece, ei fiind mici și lenți, nu puteau traversa aceste autostrăzi fără ca viața lor să nu fie în pericol.

Piticii au făcut rost de proiectele realizate de oameni și acum au nevoie de ajutorul vostru. Ei doresc să se viziteze și în viitor, deci nu le convin deloc acele autostrăzi care împart mulțimea caselor lor în două părți nevide. După ce vor afla care sunt acele autostrăzi care lor nu le plac, ei vor împiedica pe oameni prin vrăjile lor să le construiască.

Se dă numărul N, reprezentând numărul punctelor (caselor) din plan și mai multe linii drepte (autostrăzi). Determinați, corespunzător fiecărei linii, dacă cele N puncte se situează în același semiplan determinat de linia dată, sau nu. Pentru fiecare linie prelucrată va trebui să afișați răs-

punsul imediat după prelucrare, adică înainte de citirea datelor referitoare la linia următoare. Se garantează că nici o autostradă nu trece printr-o casă.

Date de intrare și ieșire

Programul va trebui să citească datele de la *intrarea standard* (`stdin` în *C/C++*, `input` în *FreePascal*) și să scrie datele de ieșire la *ieșirea standard* (`stdout` în *C/C++*, `output` în *FreePascal*).

Prima linie a intrării conține un număr întreg N ($0 \le N \le 100000$). Pe fiecare linie i dintre următoarele N se află două numere reale x_i, y_i ($-10^9 \le x_i, y_i \le 10^9$), separate printr-un singur spațiu, care reprezintă coordonatele celei de a i-a case.

Următoarele linii conțin, fiecare patru numere reale X_1, Y_1, X_2, Y_2 ($-10^9 \le X_1, Y_1, X_2, Y_2 \le 10^9$) separate prin câte un spațiu. Aceste numere reprezintă coordonatele (X_1, Y_1) și (X_2, Y_2) a două puncte diferite, aflate pe autostradă. Corespunzător fiecărei linii de la intrare, programul vostru va trebui să afișeze pe o linie mesajul `GOOD`, dacă toate punctele date se află în același semiplan determinat de linia dată, sau `BAD`, dacă există puncte de o parte și alta a liniei. După afișarea unui răspuns la ieșirea standard va trebui să realizați operația *flush* asupra *buffer*-ului de ieșire, așa cum se va arăta în continuare.

Sistemul de evaluare va opri execuția programelor voastre după ce acesta a furnizat răspunsul aferent ultimei autostrăzi citite. Programele voastre nu trebuie să își încheie singure execuția. Se garantează că există cel mult 100000 de autostrăzi.

Instrucțiuni pentru programatorii în C/C++

Pentru citirea unei linii de la intrarea standard va trebui să folosiți o secvență de tipul:

```
double X_1,Y_1,X_2,Y_2;
scanf(" %lf %lf %lf %lf",&X_1,&Y_1,&X_2,&Y_2);
```

Observați că nu există spațiu după ultimul `%lf`.

Afișarea ieșirii corespunzătoare unei singure linii se realizează astfel:

```
printf("GOOD\n");
fflush(stdout);
```

Instrucțiuni pentru programatorii în Pascal

Pentru citirea unei linii de la intrarea standard va trebui să folosiți o secvență de tipul:

```
var X_1,Y_1,X_2,Y_2:Double;
Read(X_1,Y_1,X_2,Y_2);
```

Afișarea ieșirii corespunzătoare unei singure linii se realizează astfel:

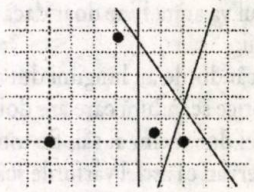
```
Writeln('GOOD');
Flush(Output);
```

Atenție!!!

Sunteți sfătuiți să folosiți tipul de date `double` (atât în *C/C++*, cât și *FreePascal*) pentru a reține numerele reale. Rețineți că în aritmetica numerelor reale pot să apară erori de rotunjire. Egalitatea a două numere reale x și y o veți verifica nu prin a testa dacă $x = y$, ci prin a testa dacă este satisfăcută relația $|x - y| < \varepsilon$ (unde ε este o constantă mai mică sau egală cu 10^{-4}).

Exemplu

Intrarea standard	*Ieșirea standard*
4	GOOD
0.0 0.0	BAD
6.00 -0.001	BAD
3.125 4.747	
4.747 0.47	
5 3 7 0	
4 -4.7 7 4.7	
4 47 4 94	



Timp maxim de execuție/test: 5 secunde
Memorie disponibilă: 16 MB

P060211: Gărzile regale

A fost o dată ca niciodată un regat care avea un rege cam paranoic. Castelul său avea fundația de formă dreptunghiulară care fusese împărțită în $M \times N$ pătrate având latura egală cu unitatea. Unele pătrate corespundeau unor ziduri, altele erau libere și corespundeau unor camere. Din cauza paranoii, regele a hotărât ca în anumite camere să amplaseze capcane ascunse.

Dar tot nu era mulțumit. A decis să așeze în castel cât mai mulți paznici posibil. Dar acest fapt nu era un lucru simplu de rezolvat. Paznicii au fost antrenați astfel încât să împuște orice persoană imediat după ce au zărit-o. Rezultă că regele trebuia să-și amplaseze paznicii cu grijă, deoarece dacă doi paznici se zăreau unul pe celălalt, ei se împușcau între ei. Evident, regele nu putea să amplaseze paznici în camerele cu capcane.

Din cauză că doi paznici aflați într-o aceeași cameră se vedeau, rezultă că într-o cameră se putea afla un singur paznic. Doi paznici aflați în camere diferite se zăreau dacă și numai dacă pătratele corespunzătoare camerelor în care se aflau erau pe aceeași linie sau pe aceeași coloană a planului castelului și dacă nu exista zid între ei.

Un paznic poate supraveghea patru direcții conform modului de deplasare a turei pe tabla de șah.

Determinați numărul maxim de paznici care se pot amplasa în interiorul castelului (conform regulilor de mai sus) și dați o posibilă amplasare a lor în camerele castelului.

Date de intrare

Pe prima linie a fișierului de intrare **`GUARDS.IN`** se află două numere M și N ($1 \le M, N \le 200$), reprezentând dimensiunile planului fundației castelului. Fiecare a i-a linie dintre următoarele M conține N numere $a_{i,1}, \ldots, a_{i,N}$, separate printr-un singur spațiu, unde:

- $a_{i,j} = 0$ indică faptul că pătratul de coordonate (i, j) este liber (corespunde unei camere fără capcană);

- $a_{i,j} = 1$ indică faptul că pătratul de coordonate (i, j) conține o capcană;
- $a_{i,j} = 2$ indică faptul că pătratul de coordonate (i, j) reprezintă un zid.

Prima coordonată a pătratului reprezintă linia, iar cea de a doua reprezintă coloana.

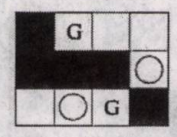
Date de ieșire

Prima linie a fișierului de ieșire **GUARDS.OUT** va trebui să conțină cel mai mare număr K de paznici care pot fi amplasați în interiorul castelului. Pe următoarele K linii va trebui să se afișeze o amplasare posibilă a celor K paznici în camere libere astfel încât doi paznici să nu se poată zări unul pe altul.

Mai exact, cea de a i-a linie dintre cele K va trebui să conțină două numere întregi r_i și c_i, separate printr-un singur spațiu, care reprezintă coordonatele camerei în care se află al i-lea paznic (linia este specificată prin r_i, iar coloana prin c_i).

Exemplu

GUARDS.IN	GUARDS.OUT
3 4	2
2 0 0 0	1 2
2 2 2 1	3 3
0 1 0 2	



Timp maxim de execuție/test: 15 secunde
Memorie disponibilă: 8 MB

P060212: Petrecere aniversară

Se apropie ziua de naștere a lui *John* și, ca în fiecare an, el dorește să organizeze un mare chef. Ar dori ca toți prietenii săi să vină la petrecere, dar constată supărat că acest lucru este aproape imposibil. De exemplu, *Susie* l-a părăsit pe *Steve* săptămâna trecută, deci este aproape imposibil să-i aibă pe amândoi alături la chef.

John a pierdut mult timp vizitând prietenii și rugându-i să vină la petrecere. A primit o serie de promisiuni, dar și mai multe rugăminți.

"*Dacă mă inviți pe mine, trebuie să-l inviți și pe prietenul meu!*" a precizat *Veronica*.

"*Dacă inviți gemenii Burdiliak, nu te aștepta ca eu sau Joseph să venim!*" l-a avertizat *Peter*.

Brusc, *John* și-a dat seama, că va fi foarte greu să satisfacă toate solicitările primite.

Veți primi o descriere a solicitărilor pe care *John* le-a primit de la prietenii săi. Va trebui să determinați un grup de persoane astfel încât, dacă *John* invită persoanele aparținând acestui grup (și pe nimeni altul), toate solicitările adunate să fie satisfăcute.

Solicitările sunt descrise în felul următor:

- `name` este o solicitare; această solicitare va fi satisfăcută dacă și numai dacă *John* îl invită pe `name` (`name` este un șir care conține cel mult 20 de litere mici ale alfabetului englez);
- `-name` este o solicitare; această solicitare va fi satisfăcută dacă și numai dacă *John* nu îl invită pe `name` (`name` este un șir care conține cel mult 20 de litere mici ale alfabetului englez);
- dacă `R1`, ..., `Rk` sunt solicitări, atunci (`R1 & ... & Rk`) este o solicitare; această solicitare va fi satisfăcută dacă și numai dacă toate solicitările `R1`, ..., `Rk` sunt satisfăcute;
- dacă `R1`, ..., `Rk` sunt solicitări, atunci (`R1 | ... | Rk`) este o solicitare; această solicitare va fi satisfăcută dacă și numai dacă cel puțin una dintre solicitările `R1`, ..., `Rk` este satisfăcută;
- dacă `R1` și `R2` sunt solicitări, atunci (`R1 => R2`) este o solicitare. Această solicitare ***nu*** va fi satisfăcută dacă și numai dacă solicitarea `R1` este satisfăcută și solicitarea `R2` nu este satisfăcută.

Date de intrare

Concurenții au primit zece fișiere de intrare, denumite **PARTY1.IN**, **PARTY2.IN**, ..., **PARTY10.IN**. Fiecare dintre acestea conține un set de date de intrare.

Pe prima linie a unui fișier de intrare se află numărul F al prietenilor lui *John*, pe următoarele F linii se află numele prietenilor, câte unul pe o linie.

Pe următoarea linie se află numărul N al solicitărilor. Următoarele N linii conțin fiecare câte o solicitare de forma celor cinci specificate în enunț.

Date de ieșire

Concurenții au furnizat zece fișiere de ieșire, denumite **PARTY1.OUT**, **PARTY2.OUT**, ..., **PARTY10.OUT**. Prima linie a unui fișier de ieșire corespunzător unui set de date (fișier de intrare) ar trebui să conțină numărul K al persoanelor pe care le va invita *John*. Următoarele K linii ar trebui să conțină numele acestora, câte unul pe linie. Se garantează existența unei soluții (nu neapărat unice) pentru fiecare fișier de intrare. Dacă există mai multe soluții, trebuie furnizată doar una dintre ele.

Exemplu

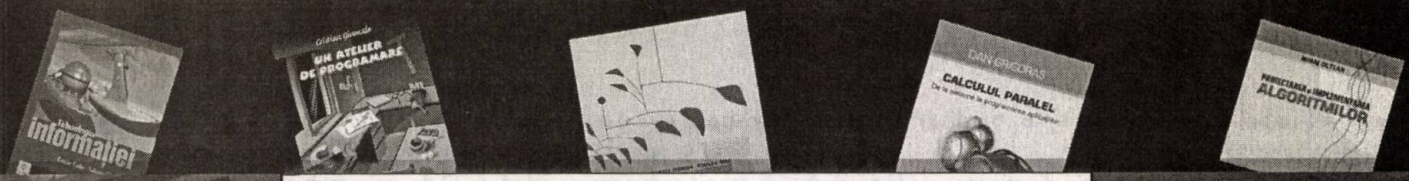
PARTY0.IN

```
3
veronica
steve
dick
3
(veronica => dick)
(steve => -veronica)
(steve & dick)
```

PARTY0.OUT

Observație

Pentru fiecare dintre cele zece fișiere de intrare, trebuia construit câte un fișier de ieșire. La sfârșitul concursului, concurenții au trimis aceste fișiere, la fel cum au trimis codurile sursă pentru celelalte probleme. Ei nu erau obligați să scrie un program dacă reușeau să genereze fișierele de ieșire corecte prin alte metode.



COMPUTER LIBRIS AGORA

Cartea este o culegere de studii de caz, construite pe temelia problemelor concursurilor Southeastern European Regional Programming Contest din perioada 1995-1999, concursuri organizate sub egida Association for Computing Machinery.
Cartea se adresează atât începătorilor, cât și celor cu experiență în domeniul programării, în încercarea de a contura prin exemple imaginea unei lumi diverse și pasionante, dar pretențioase, aceea a înzestrării mașinii de calcul cu capacitatea de a rezolva corect și eficient.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

Yong-In, Coreea de Sud

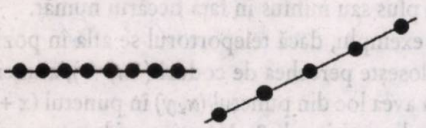
IOI 2002

Cel mai important concurs internațional de programare pentru liceeni s-a desfășurat în orașul sud-coreean Yong-In în cursul lunii august. Vă prezentăm acum enunțurile problemelor propuse spre rezolvare la a paisprezecea ediție a Olimpiadei Internaționale de Informatică.

P060213: Broscuțele buclucașe

În *Coreea*, *cheonggaeguri*-i, niște broscuțe mici, au devenit legendari pentru neastâmpărul lor. Această reputație este bine-meritată deoarece broscuțele sar prin lanurile de orez, culcând la pământ plantele. Dimineața, după ce sunt studiate plantele culcate la pământ, se dorește identificarea broscuței care a produs cele mai mari daune. O broscuță va sări întotdeauna în linie dreaptă, iar salturile au aceeași lungime.

Două broscuțe diferite pot efectua salturi de lungimi diferite sau în direcții diferite.



Lanul de orez are plantele aranjate într-un caroiaj asemănător cu cel din figura 1 și fiecare broscuță intră și iese din lan, după cum se poate vedea în figura 2.

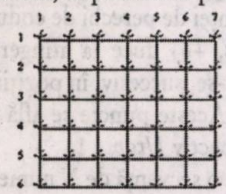


Fig. 1: Un lan de orez

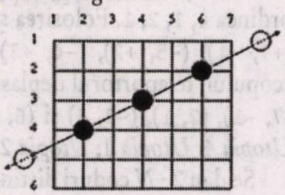


Fig. 2: Drumul unei broscuțe

Multe broscuțe pot trece prin lanul de orez, sărind de la o plantă la alta. Fiecare aterizare are loc pe o plantă care este culcată la pământ, după cum se poate vedea în figura 3. Se observă că o plantă poate fi folosită ca loc de aterizare de mai multe broscuțe. Bineînțeles, dimineața nu pot fi observate traseele broscuțelor și salturile în afara lanului, dar poate fi cercetată situația în care a ajuns lanul, așa cum se vede în figura 4.

Folosind figura 4, pot fi reconstituite toate traseele posibile pe care le-au urmat broscuțele prin lanul de orez. Ne interesează doar broscuțele care, pe parcursul traversării lanului, au aterizat pe cel puțin trei plante. Un astfel de

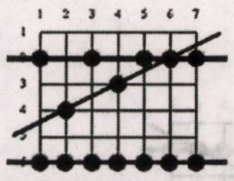


Fig. 3: Mai multe drumuri

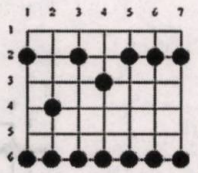


Fig. 4: Un lan dimineața

traseu este numit ***traseu al broscuței.*** Așadar, cele trei trasee prezentate în figura 3 pot fi considerate a fi trasee ale broscuțelor.

Pot fi identificate și alte trasee posibile ale broscuțelor. S-ar putea ca traseul vertical care urmează prima coloană a caroiajului să fi fost urmat de o broscuță al cărei salt are lungimea 4. Totuși, broscuța respectivă a aterizat doar pe două plante, deci nu ne interesează un astfel de traseu.

Traseul pe diagonală care conține plantele distruse de pe pozițiile de coordonate (2, 3), (3, 4) și (6, 7) conține trei plante distruse, dar broscuța ar fi trebuit să efectueze salturi de lungimi diferite, deci nici un astfel de traseu nu ne interesează.

De asemenea, se observă că, pe un traseu parcurs de o broscuță pot exista plante distruse pe care broscuța nu a aterizat. Un exemplu în acest sens este planta din poziția (2, 6) care se află pe traseul orizontal de pe a doua linie (care este parcurs de o broscuță).

De fapt, distrugerea anumitor plante s-ar putea să nu poată fi explicată de nici unul dintre traseele broscuțelor.

Va trebui să scrieți un program care determină numărul maxim de aterizări necesare pentru parcurgerea unuia dintre traseele broscuțelor. Pentru figura 4, răspunsul este 7 și traseul corespunzător este cel care parcurge a șasea linie.

Date de intrare

Programul vostru va citi datele de intrare de la *intrarea standard*. Prima linie conține două numere întregi R și C, ($1 \le R, C \le 5000$), reprezentând numărul liniilor, respectiv al coloanelor, din lanul de orez. Cea de-a doua linie conține un singur număr întreg N ($3 \le N \le 5000$), care reprezintă numărul plantelor culcate la pământ.

Fiecare dintre următoarele N linii conține două numere întregi a ($1 \le a \le R$) și b ($3 \le b \le C$), separate printr-un singur spațiu, care reprezintă linia și coloana pe care se află una dintre plantele distruse. Fiecare plantă distrusă apare o singură dată.

Date de ieșire

La *ieșirea standard* se va scrie o singură linie pe care se va afla un singur număr întreg care reprezintă numărul de plante culcate la pământ de broscuța care a produs cele mai mari daune. Dacă nu poate fi identificat nici un traseu al broscuței, se va scrie valoarea 0.

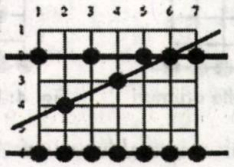
Exemple

Intrare

```
6 7
14
2 1
6 6
4 2
2 5
2 6
2 7
3 4
6 1
6 2
2 3
6 3
6 4
6 5
6 7
```

Ieșire

```
7
```

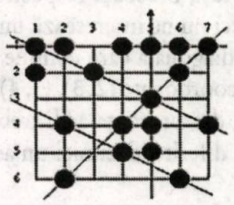


Intrare

```
6 7
18
1 1
6 2
3 5
1 5
4 7
1 2
1 4
1 6
1 7
2 1
2 3
2 6
4 2
4 4
4 5
5 4
5 5
6 6
```

Ieșire

```
4
```



Timp maxim de execuție/test: 2 secunde
Memorie disponibilă: 64 MB

P060214: Utopia divizată

Frumosul ținut *Utopia* a fost afectat de război. După încheierea ostilităților, țara a fost împărțită în patru regiuni printr-o linie verticală (pe direcția nord-sud) și una orizontală (pe direcția est-vest). Intersecția acestor două linii a ajuns să fie cunoscută sub denumirea de ***punctul (0, 0)***. Toate cele patru regiuni formate au avut pretenția să folosească denumirea *Utopia* dar, pe măsura trecerii timpului, au ajuns să fie cunoscute drept *Utopia* 1 (regiunea din nord-est), 2 (regiunea din nord-vest), 3 (regiunea din sud-vest) și 4 (regiunea din sud-est). Orice punct din interiorul oricăreia dintre cele patru regiuni a ajuns să fie identificat prin distanța spre est și distanța spre nord față de punctul (0,0). Evident, aceste distanțe pot fi negative; ca urmare, un punct din *Utopia* 1 a ajuns să fie identificat printr-o pereche (*pozitiv*, *pozitiv*), unul din *Utopia* 2 printr-o pereche (*negativ*, *pozitiv*), unul din *Utopia* 3 printr-o pereche (*negativ*, *negativ*), iar unul din *Utopia* 4 printr-o pereche (*pozitiv*, *negativ*).

Utopia 2 (–,+)	*Utopia* 1 (+,+)
Utopia 3 (–,–)	*Utopia* 4 (+,–)

O problemă majoră a apărut datorită faptului că cetățenilor nu le era permis să traverseze frontierele. Din fericire, câțiva participanți la *IOI* veniți din *Utopia* au descoperit un mijloc simplu de teleportare. Dispozitivul necesită folosirea unor coduri (reprezentate de numere) și fiecare cod poate fi utilizat o singură dată.

Provocarea în fața căreia se află membrii echipei (și dumneavoastră) este de a ghida teleportorul din punctul (0, 0) care este poziția sa inițială, prin regiunile *Utopiei* într-o ordine specificată. Nu contează unde în interiorul unei regiuni ajunge teleportorul, dar veți avea o secvență de N numere care identifică regiuni în care teleportorul trebuie să ajungă.

S-ar putea să vi se ceară să ajungeți în aceeași regiune la două sau mai multe opriri consecutive. După părăsirea poziției inițiale, teleportorul nu are voie să ajungă pe o linie de graniță.

Veți primi o secvență de $2 \cdot N$ coduri pe care va trebui să le scrieți ca o secvență de N perechi de coduri, plasând semnul plus sau minius în fața fiecărui număr.

De exemplu, dacă teleportorul se află în poziția (x, y) și se folosește perechea de coduri $(+u, -v)$, atunci teleportarea va avea loc din punctul (x, y) în punctul $(x + u, y - v)$. Aveți la dispoziție cele $2 \cdot N$ numere și le puteți utiliza în ce ordine doriți, folosind în fața lor semnul pe care îl doriți.

Să presupunem că aveți la dispoziție codurile 7, 5, 6, 1, 3, 2, 4, 8 și trebuie să ghidați teleportorul prin regiuni în ordinea 4, 1, 2, 1. Folosirea secvenței de perechi de coduri (+7, -1), (-5, +2), (-4, +3), (+8, +6) duce la atingerea scopului, teleportorul deplasându-se, succesiv, în pozițiile (7, -1), (2, 1), (-2, 4) și (6, 10). Aceste puncte se află în *Utopia* 4, *Utopia* 1, *Utopia* 2, respectiv *Utopia* 1.

Se dau $2 \cdot N$ coduri distincte și o secvență de N numere care indică regiunile în care va trebui să ajungă teleportorul. Construiți o secvență de perechi de coduri, folosind codurile date, astfel încât teleportorul să treacă prin cele N regiuni, în ordinea dată.

Date de intrare

Datele de intrare vor fi citite de la *intrarea standard*. Prima linie va conține un număr întreg pozitiv N ($1 \le N \le 10000$). A doua linie va conține $2 \cdot N$ coduri, reprezentate prin numere întregi cuprinse între 1 și 100000. Ultima linie conține o secvență de N numere cuprinse între 1 și 4, care reprezintă regiunile în care trebuie să ajungă teleportorul. Numerele de pe aceeași linie sunt separate printr-un singur spațiu.

Date de ieșire

La ieșirea standard vor fi scrise N linii, fiecare conținând o pereche de coduri; fiecare cod va fi precedat de un semn (plus sau minus). Acestea sunt perechile de coduri care vor ghida teleportorul prin regiunile Utopiei, respectând ordinea dată. Cele două coduri de pe o linie vor fi separate printr-un singur spațiu și nu pot exista spații între semne și numere.

Dacă există mai multe soluții, va trebui generată una singură. Dacă nu există nici o soluție, la ieșirea standard se va scrie doar valoarea `0`.

Exemple

Intrare	*Ieșire*
4 7 5 6 1 3 2 4 8 4 1 2 1	+7 -1 -5 +2 -4 +3 +8 +6

Intrare	*Ieșire*
4 2 5 4 1 7 8 6 3 4 2 2 1	+3 -2 -4 +5 -6 +1 +8 +7

Timp maxim de execuție/test: 2 secunde

Memorie disponibilă: 32 MB

P060215: Compresie XOR

Trebuie să implementați o aplicație pentru un tip de telefon mobil care are un ecran alb-negru. Coordonatele orizontale (x) ale ecranului încep din partea stângă, iar coordonatele verticale (y) din partea de sus. Pentru aplicație aveți nevoie de diferite imagini care nu au aceleași dimensiuni. Pentru a nu fi nevoiți să stocați imaginile, vreți să le creați folosind biblioteca grafică a telefonului. Puteți presupune că înainte să începeți desenarea primei imagini, toți pixelii de pe ecran sunt albi. Singura operație grafică disponibilă este `XOR(L,R,T,B)`, al cărei efect este inversarea valorilor din dreptunghiul care are colțul din stânga-sus în poziția `(L,T)` și cel din dreapta-jos în poziția `(R,B)`.

De exemplu, să considerăm imaginea din figura 3; vom prezenta modul în care ar putea fi obținută aceasta. Prin aplicarea operației `XOR(2,4,2,6)` asupra imaginii inițiale, se obține imaginea din figura 1. Apoi, prin aplicarea operației `XOR(3,6,4,7)` asupra imaginii obținute, se va ajunge la cea din figura 2. Pentru a obține imaginea din figura 3 va trebui să aplicăm operația `XOR(1,3,3,5)` asupra imaginii din figura 2.

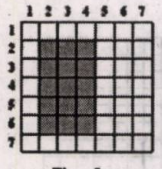


Fig. 1: XOR(2,4,2,6)

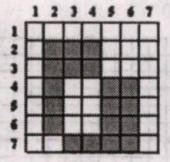


Fig. 2: XOR(3,6,4,7)

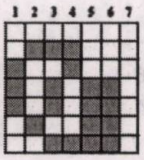


Fig. 3: XOR(1,3,3,5)

Dându-se mai multe imagini alb-negru, trebuie să generați fiecare poză pornind de la un ecran alb și folosind cât mai puține operații `XOR`. Vi se pun la dispoziție toate fișierele de intrare care descriu imaginile și trebuie să obțineți fișierele de ieșire corecte, eventual scriind un program.

Date de intrare

Există zece seturi de date, în fișierele **`XOR1.IN`**, **`XOR2.IN`**, ..., **`XOR10.IN`**. Prima linie a unui fișier de intrare conține un număr întreg N ($5 \le N \le 2000$) care indică faptul că imaginea are N linii și N coloane. Restul liniilor conțin informații referitoare la imaginea care trebuie obținută. Fiecare linie conține N numere întregi care reprezintă valorile pixelilor de pe linia repectivă. Numerele pot avea valoarea `0` (reprezintă un pixel alb) sau `1` (reprezintă un pixel negru).

Date de ieșire

Pentru fiecare fișier de intrare va trebui să creați un fișier de ieșire cu dimensiunea mai mică decât 1 MB. Prima linie a unui fișier de ieșire va fi `#FILE xor I`, unde `I` reprezintă numărul de ordine al fișierului de intrare corespunzător. Cea de-a doua linie va conține numărul K care reprezintă numărul de operații `XOR` descrise în fișier. Următoarele K linii vor reprezenta câte o operație `XOR`, în ordinea în care acestea sunt executate. Fiecare linie va conține patru numere întregi care reprezintă parametrii `L`, `R`, `T` și `B` ai operației.

Exemplu

`XOR0.IN`	**`XOR0.OUT`**
7 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 1 0 1 1 0 0 1 0 0 1 1 0 0 0 1 1 1 1 0	#FILE xor 0 3 2 4 2 6 3 6 4 7 1 3 3 5

P060216: Programarea pachetelor

Un calculator trebuie să proceseze o secvență de N operații. Operațiile sunt numerotate de la 1 la N, astfel încât secvența este 1, 2, ..., N. Secvența de operații trebuie partiționată în unul sau mai multe pachete, fiecare pachet fiind format din operații consecutive în secvență. Procesarea începe la momentul `0`.

Pachetele sunt procesate unul câte unul, începând cu primul pachet și respectând regulile care vor fi prezentate în continuare. Dacă un pachet b conține operații care au numere de ordine mai mici decât operațiile din pachetul c, atunci pachetul b este procesat înaintea pachetului c. Operațiile dintr-un pachet sunt procesate succesiv. Imediat după procesarea tuturor operațiilor dintr-un pachet, calculatorul furnizează datele de ieșire corespunzătoare tuturor operațiilor din pachet. ***Timpul de ieșire*** al unei operații j este dat de momentul în care se termină procesarea pachetului care conține operația j.

Un *timp de setare* S este necesar pentru inițializarea calculatorului pentru fiecare pachet. Pentru fiecare operație i, se cunoaște *factorul de cost* F_i și *timpul de execuție* T_i necesare pentru procesarea sa. Dacă un pachet conține operațiile x, $x + 1$, ... , $x + k$ și procesarea sa începe la momentul t, atunci timpul de ieșire ale tuturor operațiilor din pachet este $t + S + (T_x + T_{x+1} + ... + T_{x+k})$. Rețineți faptul că datele de ieșire corespunzătoare tuturor operațiilor dintr-un pachet sunt furnizate simultan. Dacă timpul de ieșire al unei operații i este O_i, *costul* său este $O_i \cdot F_i$.

De exemplu, să presupunem că avem 5 operații și timpul de setare este $S = 1$. Timpii de execuție ai operațiilor sunt $(T_1, T_2, T_3, T_4, T_5) = (1, 3, 4, 2, 1)$, iar factorii de cost sunt $(F_1, F_2, F_3, F_4, F_5) = (3, 2, 3, 3, 4)$. Dacă operațiile sunt partiționate în trei pachete ({1, 2}, {3} și {4, 5}), atunci timpii de ieșire ai operațiilor sunt $(O_1, O_2, O_3, O_4, O_5) = (5, 5, 10, 14, 14)$, iar costurile operațiilor sunt $(C_1, C_2, C_3, C_4, C_5) = (15, 10, 30, 42, 56)$.

Costul total corespunzător unei partiționări este dat de suma costurilor tuturor operațiilor. Pentru acest exemplu, costul total este 153.

Trebuie să scrieți un program care, dându-se timpul de setare, o secvență de operații, timpii și factorii lor de cost, calculează costul total minim.

Date de intrare

Programul vostru va citi datele de intrare de la *intrarea standard*. Prima linie conține numărul N al operațiilor ($1 \le N \le 10000$). Cea de-a doua linie va conține timpul de setare S, care este un număr întreg astfel încât $0 \le S \le 50$. Următoarele N linii conțin informațiile referitoare la operațiile 1, 2, ..., N, în această ordine. Liniile conțin câte două numere, separate prin câte un spațiu. Primul dintre ele este timpul de execuție T_i al operației ($1 \le T_i \le 100$), iar al doilea este factorul de cost F_i al acesteia ($1 \le F_i \le 100$).

Date de ieșire

La *ieșirea standard* se va scrie o singură linie pe care se va afla un singur număr care reprezintă costul total minim care poate fi obținut.

Exemple

Intrare	*Ieșire*
2 50 100 100 100 100	45000

Intrare	*Ieșire*
5 1 1 3 3 2 4 3 2 3 1 4	153

Timp maxim de execuție/test: 0,1 secunde
Memorie disponibilă: 32 MB

P060217: Stații de autobuz

În orașul *Yong-In* se plănuiește construirea unei rețele de autobuze cu N stații. Fiecare stație este la un colț de stradă. *Yong-In* este un oraș modern, așadar harta sa este un caroiaj de pătrățele având aceleași dimensiuni.

Două dintre stații (H_1 și H_2) trebuie să fie selectate ca *stații centrale*. Aceste stații sunt legate între ele printr-o linie directă și fiecare dintre celelalte N - 2 stații vor fi conectate direct fie cu H_1, fie cu H_2, dar nu cu ambele și cu nici o altă stație.

Distanța dintre oricare două stații este dată de cea mai scurtă rută care folosește străzile. Cu alte cuvinte, distanța dintre o stație aflată în poziția (x_1, y_1) și una aflată în poziția (x_2, y_2) este dată de $|x_2 - x_1| + |y_2 - y_1|$.

Dacă stațiile A și B sunt legate de aceeași stație centrală H_1, atunci lungimea traseului de la stația A la stația B este dată de suma distanțelor dintre A și H_1 și dintre B și H_1. Dacă stațiile sunt legate de stații centrale diferite (A de stația centrală H_1 și B de stația centrală H_2), atunci lungimea traseului dintre ele este dată de suma distanțelor dintre A și H_1, H_1 și H_2 și dintre B și H_2.

Autoritățile din *Yong-In* ar dori să se asigure că fiecare locuitor va putea ajunge în orice punct al orașului cât mai repede posibil. Ca urmare, se dorește alegerea celor două stații centrale astfel încât distanța maximă dintre oricare două stații să fie cât mai mică posibil.

O alegere P a două stații centrale și stabilirea stațiilor care sunt legate de cele două stații centrale este mai bună decât o altă alegere Q dacă și numai dacă cel mai lung traseu corespunzător alegerii P este mai scurt decât cel mai lung traseu corespunzător alegerii Q. Va trebui să scrieți un program care va determina lungimea celui mai lung traseu corespunzător celei mai bune alegeri P.

Date de intrare

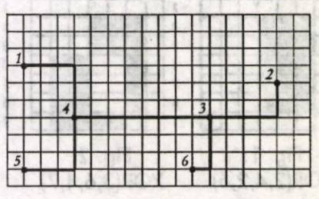
Programul vostru va citi datele de intrare de la *intrarea standard*. Prima linie va conține numărul N ($2 \le N \le 500$) al stațiilor. Fiecare dintre următoarele N linii conține coordonatele unei stații de autobuz. Aceste coordonate sunt numere întregi pozitive mai mici decât 5000. Nu pot exista două sau mai multe stații care se află în aceeași poziție.

Date de ieșire

La *ieșirea standard* se va scrie o singură linie pe care se va afla un singur număr care reprezintă cea mai mică lungime posibilă a celui mai lung traseu.

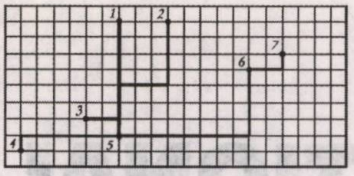
Exemple

Intrare	*Ieșire*
6 1 7 16 6 12 4 4 4 1 1 11 1	20



Intrare
7
7 9
10 9
5 3
1 1
7 2
15 6
17 7

Ieșire
25



Timp maxim de execuție/test: 4 secunde
Memorie disponibilă: 32 MB

P060218: Două bețe

Un *băț* este o secvență de cel puțin două celule consecutive ale unui caroiaj, așezate fie orizontal, fie vertical. Pe un caroiaj cu N linii și N coloane sunt așezate două bețe, unul vertical și unul orizontal. În figura 1, cele două bețe sunt reprezentate prin X-uri. Cele două bețe pot avea aceeași lungime sau, eventual, pot fi de lungimi diferite; mai mult, ele pot avea o celulă comună. Dacă, într-o diagramă cum este cea din figura 1 este posibil să interpretăm că o celulă face parte din unul din bețe sau din ambele, atunci vom considera că celula respectivă face parte din ambele bețe. De exemplu, nu se poate spune cu exactitate dacă celula (4, 4) face parte doar din bățul orizontal sau din ambele. Așadar, vom spune că cea mai de sus celulă a bățului vertical este (4, 4) și nu (5, 4).

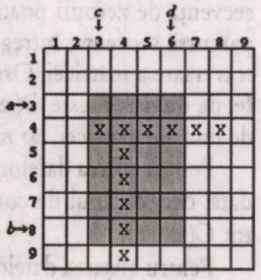


Fig. 1: Diagramă

Inițial, nu cunoaștem amplasamentul celor două bețe și sarcina dumneavoastră este de a scrie un program care determină pozițiile lor. Bățul orizontal va fi numit `BĂȚ1`, iar cel vertical `BĂȚ2`.

Fiecare celulă este reprezentată printr-o pereche de forma linie/coloană (r, c), colțul din stânga-sus al caroiajului fiind considerat (1, 1).

Fiecare băț este identificat prin cele două celule care reprezintă extremitățile sale: $\langle(r_1, c_1), (r_2, c_2)\rangle$. În figura 1 avem `ROD1 = <(4,3),(4,8)>` și `ROD2 = <(4,4),(9,4)>`.

Pentru a determina soluția va trebui să folosiți biblioteca pusă la dispoziție. Lungimea unei laturi a caroiajului este returnată de funcția `gridsize`, pe care o veți apela la început. Pentru a localiza bețele puteți folosi doar funcția `rect(a,b,c,d)`, care verifică regiunea dreptunghiulară $[a, b] \times [c, d]$ (de exemplu regiunea hașurată din figura 1), unde $a \le b$ și $c \le d$ (trebuie să fiți atenți la ordinea parametrilor). Dacă în regiunea respectivă există cel puțin o celulă care face parte dintr-un băț, atunci funcția va returna valoarea `1`; în caz contrar valoarea returnată va fi `0`. Așadar, pentru exemplul din figura 1, în urma apelului `rect(3,8,3,6)` este returnată valoarea `1`. Va trebui să scrieți un program care determină pozițiile bețelor folosind un număr limitat de apeluri ale funcției `rect`.

Rezultatul va fi furnizat folosind funcția `report(r1,c1,r2,c2,p1,q1,p2,q2)`, pusă la dispoziție de bibliotecă. Interpretarea apelului este `ROD1 = <(r1,c1),(r2,c2)>` și `ROD2 = <(p1,q1),(p2,q2)>`. Apelul acestei funcții va duce la întreruperea execuției programului.

Biblioteca

Vom prezenta în continuare modalitatea de utilizare a bibliotecii în versiunile pentru *FreePascal* și *GNU C/C++*.

Varianta FreePascal

Subprogramele care vor fi apelate sunt declarate astfel:

```
function gridsize:LongInt;
function rect(a,b,c,d:LongInt):Longint;
procedure report(r1,c1,r2,c2,p1,q1,p2,q2:
                                    Longint);
```

Codul sursă va trebui să conțină o linie de forma **`uses`** `prectlib;`.

Varianta GNU C/C++

Funcțiile care vor fi apelate sunt declarate astfel:

```
int gridsize();
int rect(int a,int b,int c,int d);
void report(int r1,int c1,int r2,int c2,
                int p1,int q1,int p2,int q2);
```

Codul sursă va trebui să conțină o linie de forma `#include "crectlib.h"`.

Restricții

- Puteți accesa datele de intrare doar prin intermediul funcțiilor `gridsize` și `rect`.
- Numărul liniilor și coloanelor din caroiaj este cuprins între 5 și 10000.
- Funcția `rect` poate fi apelată de cel mult 400 de ori. Dacă această funcție este apelată de peste 400 de ori, execuția programului va fi încheiată.
- Programul dumneavoastră va trebui să apeleze funcția `rect` de mai multe ori, în timp ce funcția `report` trebuie apelată o singură dată.
- Dacă un apel al funcției `rect` nu este valid, execuția programului va fi încheiată.
- Nu este permis ca programul dumneavoastră să citească date din fișiere, să scrie date în fișiere sau să folosească intrarea sau ieșirea standard.

Punctare

Dacă programul dumneavoastră furnizează rezultatul corect, atunci punctajul obținut depinde de numărul de apeluri ale funcției `rect`. Pentru fiecare test, veți primi întregul punctaj dacă sunt folosite cel mult 100 de apeluri. În cazul în care sunt folosite între 101 și 200 de apeluri, atunci veți primi 60% din punctaj. Pentru un număr de apeluri cuprins între 201 și 400, se va acorda 20% din punctaj.

Timp maxim de execuție/test: 1 secundă
Memorie disponibilă: 32 MB

ONI 2002, Brăila

Olimpiada NAȚIONALĂ de Informatică

Vă prezentăm în continuare soluțiile problemelor propuse la ONI 2002 la clasele a IX-a și a X-a. Aceste soluții au fost realizate de redacția GInfo pe baza soluțiilor oficiale prezentate de autorii problemelor și au fost verificate folosind seturile de date cu care au fost testate soluțiile concurenților.

P050201: Pentagon

Rezolvarea acestei probleme este foarte simplă, ea neimplicând decât parcurgerea pe coloane a unei matrice, contorizarea unor secvențe de zerouri și păstrarea unei statistici referitoare la aceste secvențe.

Pentru a determina blocurile necesare, vom determina blocurile pe fiecare coloană (datorită faptului că lățimea unui bloc este întotdeauna 1 și toate blocurile sunt dispuse pe verticală, un bloc poate ocupa o singură coloană).

Pentru a determina blocurile de pe o coloană va trebui să determinăm secvențele de zerouri de pe coloana respectivă. Vom lua în considerare doar secvențele de lungime maximă pentru a minimiza numărul total de blocuri. De exemplu, dacă avem șase zerouri consecutive, am putea folosi un bloc de lungime 6, dar și două blocuri de lungime 5 și 1, 4 și 2 etc. Evident, este obligatoriu să folosim un singur bloc pentru ca numărul total al blocurilor utilizate să fie minim.

Așadar, pentru fiecare coloană vom determina lungimile secvențelor de zerouri. O secvență de zerouri poate începe fie pe prima linie a coloanei, fie în momentul în care întâlnim o linie pe care se află un element cu valoarea 0 în timp ce pe linia anterioară se află un element cu valoarea 1. Secvența se va termina fie la terminarea parcurgerii coloanei (se ajunge pe ultima linie a acesteia), fie în momentul în care întâlnim o linie pe care se află un element cu valoarea 1 în timp ce pe linia anterioară se află un element cu valoarea 0.

În momentul detectării terminării unei secvențe (presupunem că lungimea acesteia este x), numărul blocurilor de lungime x este incrementat. Pentru păstrarea numărului de blocuri se utilizează un șir a, unde a_x indică numărul blocurilor de lungime x.

La sfârșit, vom afișa statistica cerută pe baza datelor păstrate în șirul a. Vor fi afișate toate perechile de forma i a_i care respectă condiția $a_i \neq 0$.

Analiza complexității

Datorită faptului că numărul de elemente care compun o secvență de zerouri poate fi determinat pe măsură ce este parcursă secvența, întregul algoritm constă într-o singură traversare a matricei. Ordinul de complexitate al unei astfel de traversări este $O(m \cdot n)$, unde m reprezintă numărul de linii ale matricei, iar n reprezintă numărul de coloane.

Pentru citirea datelor, matricea este parcursă o singură dată, deci ordinul de complexitate al acestei operații este tot $O(m \cdot n)$.

Pentru afișarea datelor se parcurge o singură dată șirul a; acesta nu poate conține mai mult de m elemente deoarece nu pot fi folosite blocuri mai înalte decât înălțimea zidului. Așadar, scrierea soluției are ordinul de complexitate $O(m)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(m \cdot n) + O(m \cdot n) + O(m) = O(m \cdot n)$.

Dificultate 12

Algoritmi	★☆☆☆☆
Structuri de date	★☆☆☆☆
Originalitate	★☆☆☆☆
Cunoștințe	★☆☆☆☆
Implementare	★☆☆☆☆

www.ginfo.ro/revista/12_6/surse/pentagon.cpp

P050202: Pod

Rezolvarea problemei se bazează pe o variantă simplă a metodei programării dinamice. Se observă foarte ușor că numărul de posibilități de a ajunge pe cea de-a i-a scândură depinde doar de numărul de posibilități de a ajunge pe cele trei scânduri aflate în fața ei. Vom nota cu t_i numărul de posibilități de a ajunge pe cea de-a i-a scândură. Vom con-

sidera malul opus ca fiind cea de-a $(N + 1)$-a scândură, unde N este numărul scândurilor care formează podul. Soluția problemei va fi dată de valoarea t_{N+1}.

În cazul în care cea de-a i-a scândură lipsește, pe ea nu se poate ajunge, deci vom avea $t_i = 0$.

În cazul în care această scândură există, dar este deteriorată, pe ea se poate ajunge doar de pe scândura precedentă, deci vom avea $t_i = t_{i-1}$.

În cazul în care scândura există și nu este deteriorată, pe ea se poate ajunge de pe scândura anterioară (chiar dacă este deteriorată) sau de pe oricare dintre precedentele două (dacă nu sunt deteriorate). Pentru a exprima relația matematică pentru t_i, vom folosi funcția s_i definită prin $s_i = t_i$ dacă a i-a scândură există și nu este deteriorată și $s_i = 0$ în caz contrar. Folosind această funcție, relația este $t_i = s_{i-3} + s_{i-2} + t_{i-1}$.

Inițial, vom avea $t_0 = 1$, deoarece se poate spune că există o singură posibilitate de a ajunge pe malul pe care ne aflăm.

Datorită faptului că valorile t_i pot avea până la 80 de cifre, este necesară simularea operației de adunare pentru numere mari.

Determinarea unei soluții corecte se poate realiza destul de ușor dacă, la fiecare pas, păstrăm indicele unei scânduri anterioare de pe care se poate trece pe scândura curentă. Acest indice va fi fie cel al scândurii anterioare (dacă aceasta există și numărul posibilităților de a ajunge la ea este nenul), fie al uneia dintre precedentele două (dacă există, nu este deteriorată și numărul posibilităților de a ajunge la ea este nenul). Dacă valoarea t_{N+1} este nenulă, atunci există cu siguranță cel puțin o soluție. În final, modalitatea de traversare va fi generată cu ajutorul unei tehnici recursive foarte simple.

Analiza complexității

Operația de adunare a numerelor mari are ordinul de complexitate $O(NCif)$, unde $NCif$ este numărul de cifre al celui mai mare dintre numerele care se adună. Deoarece $NCif$ este cel mult 80, ordinul de complexitate al operației de adunare poate fi considerat a fi $O(80) = 80 \cdot O(1) = O(1)$.

Trebuie efectuate cel mult $2 \cdot N$ astfel de adunări, deci operația de determinare a valorilor t_i are ordinul de complexitate $O(N) \cdot O(1) = O(N)$.

Pentru reconstituirea drumului, la fiecare pas trebuie păstrat indicele unei scânduri precedente de pe care se poate ajunge pe scândura curentă. Există doar trei posibilități de a ajunge pe scândura curentă, deci ordinul de complexitate al acestei operații este $O(1)$. Pentru determinarea scândurii anterioare corespunzătoare fiecărei scânduri din componența podului sunt efectuate $O(N)$ astfel de operații, deci ordinul de complexitate al operației de determinare a unei modalități de traversare este $O(N)$.

Citirea datelor de intrare se realizează într-un timp cu ordinul de compelexitate $O(N)$ deoarece pot exista cel mult $N - 1$ scânduri care lipsesc și cel mult N care sunt deteriorate.

Scrierea datelor de ieșire constă în generarea unei modalități de traversare care are lungimea cel mult N și a numărului modalităților de traversare. Deoarece determinarea scândurii precedente se realizează pe baza unor indici păstrați pentru fiecare scândură în parte, ordinul de complexitate al acestei operații este $O(1)$. Datorită faptului că vor fi cel mult N astfel de determinări, ordinul de complexitate al operației de determinare a modalității de traversare este $O(N)$. Scrierea numărului posibilităților de traversare poate fi considerat a fi o operație elementară, deci are ordinul de complexitate $O(1)$. Așadar, ordinul de complexitate al operației de scriere a datelor de ieșire este $O(N) + O(1) = O(N)$.

În concluzie, algoritmul de rezolvare al acestei probleme are ordinul de complexitate $O(N) + O(N) + O(N) + O(N) = O(N)$.

Dificultate 25

Algoritmi	★★☆☆☆
Structuri de date	★☆☆☆☆
Originalitate	★☆☆☆☆
Cunoștințe	★★☆☆☆
Implementare	★★☆☆☆

www.ginfo.ro/revista/12_6/surse/pod.cpp

P050203: Suma

Pentru ca numărul termenilor să fie minim, trebuie ca suma termenilor care apar cu semn negativ să fie cât mai mică posibil.

Inițial vom presupune că nu va trebui să scădem nici o valoare și ne propunem să găsim cel mai mic număr N pentru care $S_N = \sum_{i=1}^{N} i \geq S$. Valoarea N se obține rezolvând ecuația de gradul II $S_N = S$ (vom avea $S_N \leq S + N$) și rotunjind prin adaos rădăcina pozitivă obținută. Avem:

$$\frac{N \cdot (N+1)}{2} = S$$

$$N^2 + N - 2 \cdot S = 0$$

$$N = \frac{\sqrt{1 + 8 \cdot S} - 1}{2}$$

Așadar, valoarea N este $\left\lceil \frac{\sqrt{1+8 \cdot S} - 1}{2} \right\rceil$. Datorită faptului că N este cea mai mică valoare pentru care suma atinge sau depășește valoarea S, soluția problemei nu poate fi dată de nici un număr $N' < N$. În cele ce urmează vom demonstra că soluția problemei este dată de N, $N + 1$ sau $N + 2$. Se observă că numerele S_N, S_{N+1} și S_{N+2} nu pot avea toate aceeași paritate, deoarece $N + 1$ și $N + 2$ au parități diferite. Vom nota prin D_N diferența dintre valoarea S_N și valoarea care trebuie obținută (S). Deoarece se scade aceeași valoare din S_N, S_{N+1} și S_{N+2} este evident că D_N, D_{N+1} și D_{N+2} nu pot avea toate aceeași paritate.

Soluția problemei este dată de indicele celui mai mic număr par dintre aceste trei numere; fie acest indice N'.

Suma totală a elementelor cărora trebuie să le fie modificat semnul este $D_{N'}/2$. Prin modificarea semnului unui element, valoarea expresiei scade cu dublul acestui element, deci avem $S_{N'} - D_{N'} / 2 \cdot 2 = S$.

Fie $x = D_{N'} / 2$ (suma totală a elementelor care trebuie scăzute). Deoarece $N' \le N + 2$, obținem $S_{N'} \le S_N + N + 1 + N + 2$. Folosind relația $S_N \le S + N$, se obține relația $S_{N'} \le S + 3 \cdot N + 3$. Cu alte cuvinte, avem $x \le \frac{3 \cdot N + 3}{2} \le \frac{3 \cdot N' + 3}{2}$. Așadar, valoarea x poate fi obținută alegând doar două numere cuprinse între 1 și N'. În cazul în care $x > N'$ vom scădea N' și $N' - x$, iar în caz contrar vom scădea doar x.

Analiza complexității

Valoarea N poate fi obținută folosind o simplă formulă matematică, deci ordinul de complexitate al acestei operații este $O(1)$.

Valorile S_N, S_{N+1}, S_{N+2}, D_N, D_{N+1}, D_{N+2} sunt calculate tot cu ajutorul unor formule matematice simple, așadar ordinul de complexitate al operației de determinare a acestora este tot $O(1)$.

Identificarea valorii N' se face pe baza verificării parității a cel mult trei numere, deci și această operație are ordinul de complexitate $O(1)$.

Determinarea valorii x și a celor cel mult două numere cărora li se va modifica semnul este realizată tot pe baza unor calcule simple al căror ordin de complexitate este $O(1)$.

În concluzie, ordinul de complexitate al unui algoritm eficient de rezolvare a acestei probleme este $O(1) + O(1) + O(1) + O(1) = O(1)$.

Dificultate 09

Algoritmi	★☆☆☆☆
Structuri de date	☆☆☆☆☆
Originalitate	★☆☆☆☆
Cunoștințe	★☆☆☆☆
Implementare	★☆☆☆☆

www.ginfo.ro/revista/12_6/surse/suma.cpp

P050204: Becuri

Este evident că pentru a obține prima linie a matricei nu putem decât fie să comutăm toate coloanele pe care trebuie să se afle becuri aprinse și să nu comutăm prima linie, fie să comutăm prima linie și să comutăm toate coloanele pe care trebuie să se afle becuri stinse. Analog, pentru prima coloană putem fie să comutăm toate liniile pe care trebuie să se afle becuri aprinse și să nu comutăm prima coloană, fie să comutăm prima coloană și să comutăm toate liniile pe care trebuie să se afle becuri stinse.

Astfel avem patru posibilități de a obține configurația cerută:

- comutăm liniile cărora le corespund becuri aprinse pe prima coloană și coloanele cărora le corespund becuri aprinse pe prima linie;
- comutăm liniile cărora le corespund becuri aprinse pe prima coloană și coloanele cărora le corespund becuri stinse pe prima linie;
- comutăm liniile cărora le corespund becuri stinse pe prima coloană și coloanele cărora le corespund becuri aprinse pe prima linie;
- comutăm liniile cărora le corespund becuri stinse pe prima coloană și coloanele cărora le corespund becuri stinse pe prima linie.

Vom verifica fiecare dintre aceste patru variante și apoi o vom alege pe cea care implică cele mai puține comutări de linii și coloane.

Dacă nici una dintre cele patru variante nu duce la obținerea configurației cerute, putem trage concluzia că aceasta nu poate fi obținută prin comutarea unor linii și a unor coloane.

Analiza complexității

Citirea datelor de intrare implică parcurgerea unei matrice pătratice, deci ordinul de complexitate al acestei operații este $O(N^2)$.

Verificarea fiecăreia dintre cele patru posibilități de a obține soluția implică parcurgerea primei linii și a primei coloane; ordinul de complexitate al unei astfel de parcurgere este $O(N) + O(N) = O(N)$. Pentru fiecare element este posibil ca linia/coloana corespunzătoare să fie comutată. Operația de comutare implică traversarea liniei sau coloanei, așadar are ordinul de complexitate $O(N)$. Ca urmare, ordinul de complexitate al fiecăreia dintre cele patru posibilități este $O(N) \cdot O(N) = O(N^2)$. Întreaga operație de determinare a soluției are ordinul de complexitate $4 \cdot O(N^2) = O(N^2)$.

Scrierea datelor de ieșire implică traversarea șirurilor care indică dacă o linie sau coloană a fost comutată; ordinul de complexitate al operației este $O(N)$.

În concluzie, algoritmul de rezolvare al acestei probleme are ordinul de complexitate $O(N^2) + O(N^2) + O(N) = O(N^2)$.

Dificultate 19

Algoritmi	★☆☆☆☆
Structuri de date	★☆☆☆☆
Originalitate	★☆☆☆☆
Cunoștințe	★★☆☆☆
Implementare	★☆☆☆☆

www.ginfo.ro/revista/12_6/surse/becuri.cpp

P050205: Discuri

Pentru identificarea discurilor dispensabile va trebui să determinăm atingerile dintre discurile care influențează lățimea figurii. De exemplu, dacă avem șase discuri, cu raze de `1000`, `1`, `2`, `3`, `1000`, respectiv `500`, atunci atingerile care influențează lățimea figurii sunt între primul și al cincilea disc, respectiv între al cincilea și al șaselea.

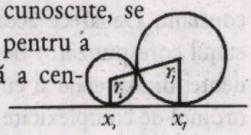
Pentru fiecare disc i vom determina, pe rând, coordonatele orizonatale pe care le-ar avea discul dacă ar atinge unul dintre discurile anterioare. În final, vom alege discul (sau axa Oy) pentru care coordonata orizontală a centrului noului disc este maximă și putem afirma că discul i ajuge în această poziție. Dacă discul i va atinge un disc anterior j, atunci discurile cu numerele de ordine cuprinse între $j + 1$ și $i - 1$ sunt dispensabile.

După ce vom lua în considerare toate cele N discuri, vom putea determina numerele de ordine ale tuturor discurilor dispensabile.

În final vom verifica dacă există discuri introduse la sfârșit care sunt dispensabile. Petntru aceasta vom determina lățimea figurii și ultimul disc care o influențează. Toate discurile introduse după acest disc sunt dispensabile.

Pentru determinarea coordonatei orizontale x_i a centrului unui disc i care atinge un disc j avem nevoie de coordonata orizontală x_j a centrului discului j, precum și de razele r_i și r_j ale celor două discuri.

Dacă aceste trei valori sunt cunoscute, se poate folosi următoarea formulă pentru a determina coordonata orizontală a centrului discului j:



$$x_j = x_i + \sqrt{(r_i + r_j)^2 - (r_i - r_j)^2}.$$

Analiza complexității

Pentru fiecare disc i care este introdus, se determină posibila coordonată x_i datorată atingerii cu toate cele $i - 1$ discuri inserate anterior. Pentru cele N discuri se vor determina, în total, $0+1+2+\text{L}+N-1=\frac{N\cdot(N-1)}{2}$ coordonate, deci ordinul de complexitate al acestei operații este $O(N^2)$.

La fiecare pas, pot fi marcate ca dispensabile cel mult toate discurile inserate anterior, așadar ordinul de complexitate al acestei operații este tot $O(N^2)$.

Determinarea lățimii figurii și a cercurilor dispensabile de la sfârșitul secvenței necesită a singură parcurgere a șirului care păstrează coordonatele centrelor discurilor, ceea ce implică ordinul de complexitate $O(N)$.

Afișarea cercurilor dispensabile precum și citirea razelor cercurilor sunt operații care se efectuează în timp liniar ($O(N)$), necesitând o simplă parcurgere a unor șiruri.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(N^2) + O(N^2) + O(N) + O(N) + O(N) = O(N^2)$.

Dificultate 10

Algoritmi	★☆☆☆☆
Structuri de date	★☆☆☆☆
Originalitate	★★☆☆☆
Cunoștințe	★☆☆☆☆
Implementare	★☆☆☆☆

www.ginfo.ro/revista/12_6/surse/discuri.cpp

P050206: Cod

Din condițiile precizate în enunț rezultă că, pe baza unei mulțimi de litere distincte, se poate construi un singur cuvânt care respectă condițiile date, și anume cel care conține literele ordonate lexicografic. Așadar, oricărei mulțimi de cel mult zece litere distincte îi corespunde un cuvânt care respectă condițiile din enunț. Se poate afirma că un cuvânt este o submulțime a mulțimii literelor; de aici rezultă că numărul cuvintelor formate din k litere este C_{26}^k. Mai mult, numărul cuvintelor format din k dintre ultimele n litere ale alfabetului este C_n^k.

Numărul de ordine al cuvântului dat este mai mare decât cel al cuvintelor formate din mai multe cifre. Din aceste motive, pentru un cunvânt format din k litere, vom avea un număr mai mare decât $\sum_{i=1}^{k} C_{26}^i$.

În continuare, pentru prima literă a cuvântului, va trebui să găsim numărul cuvintelor care încep cu o literă mai mică (din punct de vedere lexicografic). În cazul în care cuvântul are k litere, vor exista C_{25}^{k-1} cuvinte valide care încep cu `'a'`, C_{24}^{k-1} cuvinte valide care încep cu `'b'` etc. În general, vor exista C_{26-i}^{k-1} cuvinte valide care încep cu a i-a literă a alfabetului. Dacă prima literă a cuvântului este cea de-a n-a literă a alfabetului, vom avea $\sum_{i=1}^{n} C_{26-i}^{k-1}$ cuvinte valide care încep cu o literă mai mică.

În acest moment știm numărul de ordine minim al unui cuvânt care începe cu prima literă a cuvântului dat. Pentru a doua literă vom proceda într-o manieră asemănătoare. Singura diferență este dată de faptul că a doua literă trebuie să fie strict mai mare decât prima. Așadar, dacă prima literă este cea de-a n-a a alfabetului, iar a doua este cea de-a m-a, atunci vom avea $\sum_{i=n+1}^{m-1} C_{26-i}^{k-2}$ cuvinte care au pe prima poziția aceeași literă, iar pe cea de-a doua poziție o literă mai mică.

Procedeul va continua pentru fiecare literă în parte. În cazul în care litera curentă este cea de-a p-a a cuvântului, este a m-a a alfabetului, iar litera anterioară este a n-a a alfabetului, numărul de cuvite care au pe primele $p - 1$ poziții aceleași litere ca și cuvântul dat, iar pe cea de-a p-a o literă mai mică este dat de formula $\sum_{i=n+1}^{m-1} C_{26-i}^{k-p}$.

Adunând toate valorile obținute pe parcurs vom obține numărul cuvintelor care se află înaintea cuvântului dat. Adunând 1 la această valoare, vom obține numărul de ordine al cuvântului.

Analiza complexității

Pentru a analiza complexitatea acestui algoritm va trebui să precizăm faptul că numărul literelor din alfabet este

constant, deci nu poate interveni în exprimarea ordinului de complexitate. Acesta va fi stabilit doar în funcție de lungimea k a cuvântului.

Inițial se calculează suma $\sum_{i=1}^{k} C_{26}^{i}$, operație realizabilă în timp liniar având în vedere observația anterioară. Așadar, primul pas al algoritmului are ordinul de complexitate $O(k)$.

Pentru fiecare literă a cuvântului se calculează suma $\sum_{i=n+1}^{m-1} C_{26-i}^{k-2}$, unde variabilele au semnificația prezentată anterior. Numărul de litere este implicat în determinarea valorii combinării. Așadar, calculul combinării se realizează în timp liniar. Numărul de combinări calculate nu depinde de lungimea cuvântului, deci ordinul de complexitate al calculării acestei sume este $O(k)$. În total vor fi calculate k astfel de sume, deci ordinul de complexitate al celui de-al doilea pas al algoritmului este $O(k) \cdot O(k) = O(k^2)$.

Citirea datelor de intrare și scrierea celor de ieșire se realizează cu ajutorul unor operații elementare, deci putem considera că au ordinul de complexitate $O(1)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(k^2) + O(k) + O(k) = O(k^2)$ având în vedere că numărul literelor din alfabetul englez este constant.

Dificultate 21

Algoritmi	★☆☆☆☆
Structuri de date	☆☆☆☆☆
Originalitate	★☆☆☆☆
Cunoștințe	★★☆☆☆
Implementare	★☆☆☆☆

www.ginfo.ro/revista/12_6/surse/cod.cpp

P050207: Hotel

Pentru început, vom determina numărul de etaje la care există angajați și vom atribui fiecărui etaj câte o culoare. Pe măsură ce citim datele referitoare la angajați, vom verifica dacă etajul la care lucrează angajatul are asociată o culoare și, dacă este cazul, îi vom atribui o culoare și vom crește numărul etajelor distincte. Dacă ajungem în situația în care numărul culorilor disponibile este mai mic decât numărul etajelor distincte, atunci problema nu are soluție. Atribuind câte o culoare fiecărui etaj, practic, am rezolvat cea de-a doua parte a problemei.

Pentru prima parte vom considera că avem la dispoziție n culori și există k etaje distincte la care lucrează angajați. Va trebui să determinăm numărul de posibilități de a atribui celor k etaje câte o culoare astfel încât fiecare etaj să aibă propria sa culoare.

Practic, având la dispoziție o mulțime formată din n elemente va trebui să determinăm câte posibilități de alegere a k dintre aceste elemente există. Este foarte ușor de observat că, de fapt, propoziția anterioară reprezintă definiția noțiunii matematice de *aranjamente*. Așadar, soluția primei părți a problemei este dată de formula:

$$A_n^k = \frac{n!}{(n-k)!} = (n-k+1)\cdot(n-k+2)\cdot\ldots\cdot(n-1)\cdot n.$$

Datorită faptului că se obțin numere foarte mari, nu se pot folosi tipurile de date puse la dispoziție de limbajele de programare pentru operații aritmetice. Se observă că, folosind formula anterioară, avem nevoie doar de înmulțiri dintre un număr mare și un număr cel mult egal cu 200. Ca urmare, va trebui să implementăm operația de înmulțire a unui număr mare cu un scalar. Pentru a mări viteza de execuție a programului și a utiliza mai eficient memoria, se poate folosi baza 10000 pentru efectuarea calculelor.

Analiza complexității

Citirea datelor de intrare se realizează în timp liniar, deci ordinul de complexitate al acestei operații este $O(n)$.

Verificarea faptului că unui etaj îi corespunde o culoare și eventuala atribuire a unei culori se realizează în timp constant. Deoarece această operație se realizează pentru etajul corespunzător fiecărui angajat, stabilirea corespondenței dintre etaje și culori se realizează într-un timp cu ordinul de complexitate $O(n) \cdot O(1) = O(n)$.

Calcularea numărului de modalități de alegere a culorilor implică folosirea numerelor mari. Se observă că numărul de cifre (în baza 10000) al rezultatului este mai mic decât 100, așadar putem considera că o înmulțire se realizează în timp liniar. Numărul înmulțirilor efectuate este k, deci ordinul de complexitate al operației de determinare a rezultatului este $O(k)$.

Afișarea numărului de modalități este realizată în timp constant, iar operația de afișare a culorilor corespunzătoare fiecărui angajat are ordinul de complexitate $O(n)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(n) + O(n) + O(k) + O(1) + O(n) = O(k + n)$.

Dificultate 17

Algoritmi	★☆☆☆☆
Structuri de date	☆☆☆☆☆
Originalitate	★☆☆☆☆
Cunoștințe	★☆☆☆☆
Implementare	★☆☆☆☆

www.ginfo.ro/revista/12_6/surse/hotel.cpp

P050208: Lac

Pentru rezolvarea acestei probleme vom folosi o variantă puțin modificată a algoritmului lui *Lee*. Pentru fiecare poziție a matricei care reprezintă o zonă mlăștinoasă vom determina numărul minim de pontoane care trebuie să fie amplasate pentru a se ajunge în poziția respectivă.

Pentru aceasta vom considera că pentru a ajunge în imediata vecinătate a primei linii a matricei numărul de pontoane necesare este 0. Pentru o anumită poziție, numă-

rul de pontoane necesare ajungerii în punctul respectiv este dat de cel mai mic număr corespunzător uneia dintre pozițiile învecinate la care se adaugă, eventual, un ponton dacă poziția nu corespunde unei porțiuni de uscat.

După o astfel de parcurgere a matricei, vom avea pentru fiecare poziție a matricei o anumită valoare, dar nu suntem siguri că aceasta este cea minimă. Vom parcurge succesiv matricea încercând să îmbunătățim valorile obținute. O valoare va fi modificată dacă se poate ajunge în poziția curentă dintr-o poziție învecinată și se obține un număr mai mic de pontoane pentru poziția curentă. În momentul în care nu va mai exista nici o parcurgere care să aducă îmbunătățiri, am obținut rezultatul final.

Practic cele două tipuri de parcurgeri pot fi "asimilate" în una singură dacă, inițial, se marchează toate pozițiile matricei cu o valoare suficient de mare. De fiecare dată când are loc o îmbunătățire, vom păstra direcția din care s-a ajuns în poziția curentă pentru a putea reconstitui drumul.

În final, numărul minim de pontoane va fi dat de cea mai mică valoare de pe ultima linie a matricei. Pentru reconstituirea drumului, se va porni în sens invers, de pe poziția de pe ultima linie în poziția din care s-a ajuns în ea și așa mai departe, până se ajunge pe prima linie. Dacă se ajunge într-o poziție care nu corespunde unei zone de uscat, atunci va trebui amplasat un ponton în acea poziție.

Analiza complexității

Citirea datelor de intrare implică o traversare a matricei, ordinul de complexitate al acestei operații fiind $O(m \cdot n)$.

O parcurgere a matricei în vederea realizării unei îmbunătățiri are același ordin de compelxitate. Datorită faptului că, după prima parcurgere, suntem siguri că se poate ajunge pe ultima linie folosind m pontoane (completarea integrală a coloanei), nu se vor realiza în nici o situație mai mult de m parcurgeri de îmbunătățire. Din aceste motive, ordinul de complexitate al operației de determinare a numărului de pontoane necesare pentru ajungerea în fiecare poziție este $O(m) \cdot O(m \cdot n) = O(m^2 \cdot n)$.

Determinarea valorii minime de pe ultima linie se realizează în timp liniar, ordinul de complexitate fiind $O(n)$.

Reconstituirea drumului ar putea, teoretic, să necesite traversarea întregii matrice. Astfel, în cel mai defavorabil caz, ordinul de complexitate al acestei operații este $O(m \cdot n)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(m \cdot n) + O(m^2 \cdot n) + O(n) + O(m \cdot n) = O(m^2 \cdot n)$.

Dificultate 33

Algoritmi	★★★☆☆
Structuri de date	★☆☆☆☆
Originalitate	★★☆☆☆
Cunoștințe	★★☆☆☆
Implementare	★☆☆☆☆

www.ginfo.ro/revista/12_6/surse/lac.cpp

P050209: Logic

Pentru început vom observa că, pentru ca trasarea să fie posibilă, fiecare zonă trebuie să fie delimitată de un număr par de segmente. Aceasta se datorează faptului că linia trebuie să intre și să iasă din fiecare zonă de un anumit număr de ori.

Așadar, pentru fiecare zonă va trebui să numărăm segmentele (așa cum sunt definite ele în enunț) care o delimitează și să verificăm dacă numărul obținut este par sau impar. În momentul în care găsim o zonă delimitată de un număr par de segmente, vom ști că trasarea nu este posibilă. Dacă toate zonele sunt delimitate de un număr par de segmente, atunci trasarea este posibilă.

Numărarea segmentelor care delimitează o zonă se realizează pe baza unui algoritm de umplere. În momentul în care ajungem la un punct de la marginea zonei, se pune problema creșterii numărului de segmente. Există mai multe situații în care numărul de segmente va crește:

- se ajunge în partea de sus a zonei, iar punctul aflat imediat deasupra și cel din dreapta acestuia fac parte din zone diferite;
- se ajunge în partea de jos a zonei, iar punctul aflat imediat dedesubt și cel din dreapta acestuia fac parte din zone diferite;
- se ajunge în partea din stânga a zonei, iar punctul aflat imediat la stânga și cel de deasupra sa fac parte din zone diferite;
- se ajunge în partea din dreapta a zonei, iar punctul aflat imediat la dreapta și cel de deasupra sa fac parte din zone diferite;
- se ajunge într-unul din colțurile zonei.

Dacă ultima condiție este îndeplinită simultan cu una dintre primele patru, numărul segmentelor va crește doar cu 1. Creșterile datorate primelor patru condiții sunt cumulative.

Analiza complexității

Vom studia acum complexitatea algoritmului pentru un desen. Citirea datelor de intrare implică o traversare a matricei, deci această operație se realizează într-un timp de ordinul $O(n^2)$.

La fiecare pas al algoritmului de umplere se numără segmentele care trebuie adăugate pentru poziția curentă. Se pot adăuga cel mult patru segmente, așadar ordinul de complexitate al operației este $O(1)$. Algoritmul de umplere implică parcurgerea integrală a matricei, motiv pentru care vor fi vizitate toate cele n^2 poziții. Așadar, algoritmul de verificare a posibilității de trasare a liniei are ordinul de complexitate $O(n^2) \cdot O(1) = O(n^2)$.

Datele de ieșire constau din scrierea unui singur mesaj, deci această operație se efectuează în timp *liniar*.

În concluzie, algoritmul de rezolvare a acestei probleme, pentru un desen, are ordinul de complexitate $O(n^2) + O(n^2) + O(1) = O(n^2)$.

Pentru a stabili ordinul de complexitate al algoritmului de rezolvare a întregii probleme, vom considera că un set

de date cuprinde t desene. Așadar, ordinul de complexitate este $O(t) \cdot O(n^2) = O(n^2 \cdot t)$.

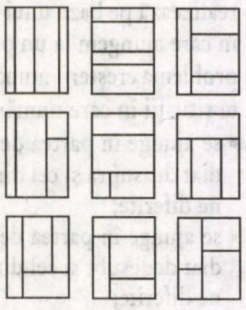
Dificultate 38

Algoritmi	★★☆☆☆
Structuri de date	★☆☆☆☆
Originalitate	★★★☆☆
Cunoștințe	★★★☆☆
Implementare	★☆☆☆☆

www.ginfo.ro/revista/12_6/surse/logic.cpp

P050210: Foto

Se observă foarte ușor că există doar nouă posibilități de lipire a patru fotografii pe o pagină, indiferent de dimensiunile acestora. Toate celelalte posibilități sunt echivalente cu una dintre cele nouă, ele obținându-se prin translatări sau oglindiri. Cele nouă aranjamente posibile pe care le vom lua în considerare sunt prezentate în figura alăturată.



Modalități de amplasare

Va trebui să alegem pozițiile celor patru fotografii pentru fiecare dintre cele nouă configurații. Teoretic, pentru fiecare dintre cele nouă configurații, există $4! = 24$ posibilități. Practic, se observă că există și în acest caz configurații echivalente. De exemplu, pentru prima și a doua configurație, nu contează ordinea în care sunt așezate fotografiile, pentru a treia configurație nu contează ordinea în care sunt dispuse cele trei poze de pe a doua coloană etc. Practic, pentru fiecare configurație vom avea 1, 2, 4, 6 sau 12 posibilități.

Cu excepția ultimei configurații, în toate celelalte există o fotografie care ocupă o întreagă latură a paginii. Cunoscând raportul dintre lungimea și lățimea sa se determină laturile spațiului liber rămas pe foaie. În continuare, una dintre laturile spațiului este ocupat integral de o fotografie. Procedura continuă până determinăm dimensiunile tuturor fotografiilor. În final se verifică dacă foaia este acoperită integral. Pentru ultima configurație vom alege, pe rând, dimensiunile posibile ale fotografiei din colțul din stânga sus și dimensiunile celorlalte fotografii vor fi determinate în același mod ca și în situațiile anterioare.

Analiza complexității

Citirea datelor se realizează în timp *constant*, deoarece numărul fotografiilor este întotdeauna 4.

Studierea primelor opt configurații se realizează de asemenea în timp constant deoarece numărul posibilităților de amplasare a fotografiilor respectând configurațiile considerate este constant.

Pentru cea de-a noua configurație putem avea cel mult $\min(X, Y)$ dimensiuni posibile ale primei fotografii, unde X și Y sunt dimensiunile paginii. Dacă notăm cu n acest minim, ordinul de complexitate al acestei operații va fi $O(n)$.

Datele de ieșire constau în exact opt numere, deci scrierea rezultatelor se realizează în timp *constant*.

În concluzie, algoritmul de rezolvare a acestei probleme are ordinul de complexitate $O(1) + 8 \cdot O(1) + O(n) + O(1) = O(n)$.

Dificultate 30

Algoritmi	★★☆☆☆
Structuri de date	★☆☆☆☆
Originalitate	★★☆☆☆
Cunoștințe	★★☆☆☆
Implementare	★★★☆☆

www.ginfo.ro/revista/12_6/surse/foto.cpp

P050211: Balanță

Se observă că suma momentelor greutăților este cuprinsă între -6000 și 6000 (dacă avem 20 de greutăți cu valoarea 20 și acestea sunt amplasate pe cel mai îndepărtat cârlig față de centrul balanței, atunci modulul sumei momentelor forțelor este $15 \cdot 20 \cdot 20 = 6000$). Ca urmare, putem păstra un șir a ale cărui valori a_i vor conține numărul posibilităților ca suma momentelor greutăților să fie i. Indicii șirului vor varia între -6000 și 6000. Pentru a îmbunătăți viteza de execuție a programului, indicii vor varia între $-300 \cdot g$ și $300 \cdot g$, unde g este numărul greutăților. Pot fi realizate îmbunătățiri suplimentare dacă se determină distanțele maxime față de mijlocul balanței ale celor mai îndepărtate cârlige de pe cele două talere și suma totală a greutăților. Dacă distanțele sunt d_1 și d_2, iar suma este s, atunci indicii vor varia între $-d_1 \cdot s$ și $d_2 \cdot s$.

Inițial, pe balanță nu este agățată nici o greutate, așadar suma momentelor greutăților este 0. Ca urmare, inițial valorile a_i vor fi 0 pentru orice indice nenul și $a_0 = 1$ (există o posibilitate ca inițial suma momentelor greutăților să fie 0 și nu există nici o posibilitate ca ea să fie diferită de 0).

În continuare, vom încerca să amplasăm greutățile pe cârlige. Fiecare greutate poate fi amplasată pe oricare dintre cârlige. Să presupunem că la un moment dat există a_i posibilități de a obține suma i. Dacă vom amplasa o greutate de valoare g pe un cârlig aflat la distanța d față de centrul balanței, suma momentelor greutăților va crește sau va scădea cu $g \cdot d$ (în funcție de brațul pe care se află cârligul). Ca urmare, după amplasarea noii greutăți există a_i posibilități de a obține suma $i + g \cdot d$. Considerăm că șirul b va conține valori care reprezintă numărul posibilităților de a obține sume ale momentelor forțelor după amplasarea greutății curente. Înainte de a testa posibilitățile de plasare a greutății, șirul b va conține doar zerouri. Pentru fiecare pereche (i, d), valoarea $b_{i+g \cdot d}$ va crește cu a_i. După considerarea tuturor perechilor, vom putea trece la o nouă greutate. Valorile din șirul b vor fi salvate în șirul a, iar șirul b

va fi reinițializat cu 0. Rezultatul final va fi dat de valoarea a_0 obținută după considerarea tuturor greutăților disponibile.

Analiza complexității

Pentru studiul complexității vom nota numărul greutăților cu g, iar cel al cârligelor cu c.

Citirea datelor de intrare corespunzătoare cârligelor și greutaților se realizează în timp liniar, deci ordinul de complexitate al acestor operații este $O(g)$, respectiv $O(c)$.

Singura mărime care nu este considerată constantă și de care depinde numărul de posibilități de a obține sumele este numărul greutăților. Așadar, vom considera că ordinul de complexitate al traversării șirului a este $O(g)$. Numărul de traversări este dat tot de numărul greutăților disponibile, deci vom avea $O(g)$ traversări. În timpul traversării vom considera toate cârligele pentru fiecare element al șirului. Ca urmare, ordinul de complexitate al operațiilor efectuate asupra unui element într-o parcurgere este $O(c)$. Rezultă că ordinul de complexitate al unei parcurgeri este $O(g) \cdot O(c) = O(g \cdot c)$, în timp ce ordinul de complexitate al întregii operații care duce la obținerea rezultatului este $O(g) \cdot O(g \cdot c) = O(g^2 \cdot c)$.

Afișarea numărului de posibilități de a echilibra balanța se realizează în timp *constant*.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(c) + O(g) + O(g^2 \cdot c) + O(1) = O(g^2 \cdot c)$.

Dificultate 23

Algoritmi	★★☆☆☆
Structuri de date	★☆☆☆☆
Originalitate	★★☆☆☆
Cunoștințe	★★☆☆☆
Implementare	★☆☆☆☆

www.ginfo.ro/revista/12_6/surse/balanta.cpp

P050212: Aliniere

Pentru fiecare soldat vom determina cel mai lung subșir strict crescător (din punct de vedere al înălțimii) de soldați care se termină cu el, respectiv cel mai lung subșir strict descrescător de soldați care urmează după el. După această operație, vom determina soldatul pentru care suma lungimilor celor două șiruri este maximă.

Chiar dacă s-ar părea că în acest mod am găsit soluția problemei, mai există o posibilitate de a mări numărul soldaților care rămân în șir. Să considerăm soldatul cel mai înalt în șirul rămas (cel căruia îi corespunde suma maximă). Acesta poate privi fie spre stânga, fie spre dreapta șirului. Din aceste motive, la stânga sau la dreapta sa poate să se afle un soldat de aceeași înălțime; unul dintre cei doi va privi spre dreapta, iar celălalt spre stânga. Totuși, nu putem alege orice soldat cu aceeași înălțime, ci doar unul pentru care lungimea șirului strict crescător (dacă se află spre stânga) sau a celui strict descrescător (dacă se află spre dreapta) este aceeași cu lungimea corespunzătoare șirului strict crescător, respectiv strict descrescător, corespunzătoare celui mai înalt soldat dintre cei rămași în șir.

După identificarea celor doi soldați de înălțimi egale (sau demonstrarea faptului că nu există o pereche de acest gen care să respecte condițiile date) se marchează toți soldații din cele două subșiruri. Ceilalți soldați vor trebui să părăsească formația.

Analiza complexității

Citirea datelor de intrare se realizează în timp liniar, deci ordinul de complexitate al acestei operații este $O(n)$.

Chiar dacă există algoritmi eficienți (care rulează în timp liniar-logaritmic) de determinare a celui mai lung subșir ordonat, timpul de execuție admis ne permite folosirea unui algoritm simplu, cu ordinul de complexitate $O(n^2)$. Acesta va fi aplicat de două ori, după care se va căuta valoarea maximă a sumei lungimilor șirurilor corespunzătoare unui soldat; așadar identificarea soldatului care poate privi în ambele direcții este o operație cu ordinul de complexitate $O(n^2) + O(n^2) + O(n) = O(n^2)$.

Urmează eventuala identificare a unui alt soldat de aceeași înălțime care respectă condițiile referitoare la lungimile subșirurilor. Pentru aceasta se parcurge șirul soldaților de la soldatul identificat anterior spre extremități; operația necesită un timp *liniar*.

Deteminarea celor două subșiruri se realizează în timp *liniar* dacă, în momentul construirii celor două subșiruri, se păstrează predecesorul, respectiv succesorul fiecărui soldat. În timpul parcurgerii subșirurilor sunt marcați soldații care rămân în formație.

Pentru scrierea datelor de ieșire se parcurge șirul marcajelor și sunt identificați soldații care părăsesc formația. Ordinul de complexitate al acestei operații este $O(n)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(n) + O(n^2) + O(n) + O(n) + O(n) = O(n^2)$.

Dificultate 33

Algoritmi	★★☆☆☆
Structuri de date	★☆☆☆☆
Originalitate	★★☆☆☆
Cunoștințe	★★★☆☆
Implementare	★☆☆☆☆

www.ginfo.ro/revista/12_6/surse/aliniere.cpp

Etapa FINALĂ

A treia ediție a concursului de programare organizat de Gazeta de Informatică s-a încheiat. Primii clasați după cele cinci runde desfășurate prin e-mail au participat la faza finală care a avut loc pe data de 1 iunie 2002 la Liceul "Emil Racoviță" din Cluj.

Dintre cei zece concurenți care s-au calificat la faza finală, doar cinci au reușit să se prezinte la concurs. Aceștia sunt:

Csaba András, Oradea
Mugurel-Ionuț Andreica, București
Adrian Balașko, Zalău
Maximilian Machedon, București
Cosmin-Silvestru Negrușeri, Bistrița

Bursa de 50$/lună oferită de editura *Agora Media* a fost câștigată, pentru al doilea an consecutiv, de *Mugurel-Ionuț Andreica*. Pe locul al doilea s-a clasat *Cosmin-Silvestru Negrușeri*, care a câștigat o bursă de 30$/lună.

În continuare vă prezentăm enunțurile celor cinci probleme propuse spre rezolvare la etapa finală a celei de-a treia ediții a concursului de programare organizat de revista noastră.

P060219: Conducte

Într-o țară există k rafinării, unde din țiței se prepară benzină și n benzinării, unde trebuie să ajungă prin conducte benzina preparată de rafinării. Se pune problema amplasării conductelor care leagă rafinăriile de benzinării, astfel încât lungimea totală a conductelor să fie minimă.

Stabiliți care sunt rafinăriile și benzinăriile între care se vor amplasa conductele, astfel încât lungimea lor totală să fie minimă.

Date de intrare

Pe prima linie a fișierului de intrare `CONDUCTE.IN` se află două numere naturale k și n, reprezentând numărul rafinăriilor, respectiv numărul benzinăriilor, separate printr-un spațiu. Următoarele k linii conțin câte două numere întregi, separate printr-un spațiu, reprezentând coordonatele rafinăriilor; următoarele n linii conțin, de asemenea, câte două numere întregi, separate printr-un spațiu, reprezentând coordonatele benzinăriilor. Coordonatele corespund unor indici de linii, respectiv de coloană într-un caroiaj imaginar care conține punctele corespunzătoare rafinăriilor și benzinăriilor.

Date de ieșire

În fișierul de ieșire `CONDUCTE.OUT` se vor scrie acele locuri care se vor lega prin conductele rețelei având lungimea totală minimă. În fiecare linie se vor scrie două numere întregi, separate printr-un spațiu, reprezentând numerele de ordine a două puncte legate. Indicele rafinăriei se va preceda de litera `'R'`, iar indicele benzinăriei de litera `'B'`.

Restricții

- $1 \le k \le 10$;
- $1 \le n \le 1000$;
- coordonatele refinăriilor și benzinăriilor sunt numere întregi din intervalul [-100, 100];
- conductele se pot ramifica doar în dreptul benzinăriilor sau în dreptul rafinăriilor;
- este posibil ca două benzinării să fie legate prin conducte (altfel spus, nu este obligatoriu ca într-o benzinărie benzina să sosească direct dintr-o rafinărie).

Exemplu

`CONDUCTE.IN`	`CONDUCTE.OUT`
`2 4`	`B2 B4`
`100 100`	`R2 B3`
`0 100`	`B2 B3`
`-100 100`	`R2 B1`
`100 0`	
`40 40`	
`100 -50`	

Timp maxim de execuție/test: 5 secunde

P060220: Dune

Pe *Câmpia Arakeenului* are loc bătălia finală pentru controlul planetei *Dune*. Câmpia este reprezentată printr-o matrice având șapte coloane și N linii.

În fiecare zonă (celulă a matricei) se află câte un contingent de trupe ale *atreizilor* și ale *harkonnienilor*. Valoarea corespunzătoare unei zone este dată de diferența dintre numărul soldaților celor două case. Valoarea este pozitivă dacă *harkonnienii* sunt mai numeroși, negativă dacă sunt

mai multe trupe *Atreides* și 0 dacă cele două contingente sunt egale.

Paul Muad'Dib are la dispoziție k bombe nucleare (*Arde-Piatră*) pe care le-a achiziționat la un preț foarte mare de la *ixieni*. Fiecare bombă poate fi lansată asupra unei secțiuni pătratice având latura 2. Așadar, patru zone vor fi afectate de un *Arde-Piatră*. Evident, toți soldații aflați în acele zone vor fi nimiciți.

Bombele trebuie lansate complet în interiorul câmpiei. *Paul* dorește ca pagubele provocate de cele k bombe să fie maxime (diferența dintre numărul *harkonnenilor* și cel al *atreizilor* din zonele afectate să fie cât mai mare). Pentru aceasta el poate folosi nici una, una, mai multe sau toate bombele pe care le-a achiziționat.

Date de intrare

Prima linie a fișierului de intrare **DUNE.IN** va conține două numere întregi, separate printr-un singur spațiu. Primul va indica numărul N al liniilor matricei, iar al doilea va indica numărul K al bombelor pe care le are la dispoziție *Paul*. Fiecare dintre următoarele N linii va conține câte șapte numere întregi separate prin câte un spațiu. Acestea vor reprezenta cele $7 \cdot N$ elemente ale matricei. Pe prima dintre aceste linii se vor afla elementele corespunzătoare primei linii a matricei, pe cea de-a doua linie cele corespunzătoare celei de-a doua linii a matricei etc. Primul element dintr-o linie va corespunde elementului de pe prima coloană a liniei respective, al doilea va corespunde elementului de pe a doua coloană etc.

Date de ieșire

Fișierul de ieșire va conține o singură linie pe care se va afla un număr întreg care va indica diferența maximă dintre numărul *harkonnienilor* și numărul *atreizilor* afectați de bombe.

Restricții

- $2 \le n \le 100$;
- $1 \le k \le 100$.

Exemplu

DUNE.IN

```
3 4
1 1 0 0 0 0 0
1 1 1 0 0 -3 0
-4 1 1 0 0 1 1
```

DUNE.OUT

```
7
```

Timp maxim de execuție/test: 1 secundă

P060221: Frăția Inelului

După ce a primit *Inelul Puterii* de la *Bilbo*, *Frodo* a fost nevoit să fugă din *Comitat* deoarece era urmărit de duhurile *Inelului*. A reușit să ajungă în *Vâlceaua Despicată*, loc în care s-a ținut marele sfat din casa lui *Elrond*.

Aici s-a hotărât ca inelul să fie distrus. Pentru aceasta el trebuia transportat la *Muntele Osândei*, singurul loc în care acesta putea fi distrus. *Frodo* a fost ales ca *Purtător al Inelului* și s-a decis ca opt reprezentanți ai semințiilor puse în pericol de *Umbra* din *Mordor* să îl însoțească. Astfel s-a format *Frăția Inelului*.

Din păcate au izbucnit certuri cu privire la traseul care urma să fie parcurs. Pentru a rezolva această problemă, *Elrond* și-a dat seama că, mai întâi, trebuie cunoscut numărul traseelor posibile. Sarcina calculării acestui număr i-a revenit lui *Sam cel Înțelept*. El a primit cea mai nouă hartă a *Pământului de Mijloc* pe care erau marcate cele mai importante puncte de reper, precum și traseele care le unesc.

Drumurile care unesc două repere sunt cu sens unic și nu există posibilitatea ca, dacă s-a ajuns la un anumit punct de reper, să existe o cale spre un punct de reper anterior. Și acum, va trebui să vă închipuiți că sunteți *Sam*, și că de dumneavoastră depinde succesul expediției.

Date de intrare

Fișierul de intrare **SAM.IN** conține, pe prima linie, numărul M al drumurilor care leagă două repere. Pe fiecare dintre următoarele M linii se vor afla câte două cuvinte (formate doar din litere mari ale alfabetului latin) care vor reprezenta reperele unite de un drum (sensul este de la primul la al doilea reper).

Date de ieșire

Fișierul de ieșire **SAM.OUT** va conține o singură linie pe care se va afla numărul traseelor.

Restricții și precizări

- există cel mult 1000 de drumuri și 100 de repere;
- numărul total al traseelor este întotdeauna cuprins între 1 și 2.000.000.000;
- se pleacă întotdeauna din VALCEAUADESPICATA;
- se ajunge întotdeauna la MUNTELEOSANDEI.

Exemplu

SAM.IN

```
9
ROHAN GONDOR
VALCEAUADESPICATA COMITAT
VALCEAUADESPICATA ROHAN
VALCEAUADESPICATA MORIA
MORIA ROHAN
MORIA GONDOR
MORIA MORDOR
GONDOR MUNTELEOSANDEI
ROHAN MUNTELEOSANDEI
```

SAM.OUT

```
5
```

Timp maxim de execuție/test: 1 secundă

P060222: Litere

Se consideră un șir de litere mari ale alfabetului latin. Acest șir se află pe prima linie a unei matrice pătratice A, având dimensiunea egală cu lungimea șirului. Următoarea linie a matricei A se obține prin permutarea la stânga cu o poziție

a elementelor de pe prima linie. Aceeași operație se aplică și pentru următoarele linii: linia i se obține prin permutarea la stânga cu o poziție a elementelor de pe linia i - 1.

În continuare se generează o matrice B care conține liniile matricei A ordonate lexicografic.

De exemplu, dacă șirul de caractere considerat este GINFO, atunci matricea A va avea următoarea structură:

```
GINFO
INFOG
NFOGI
FOGIN
OGINF
```

Ordonând lexicografic liniile matricei A, se obține matricea B:

```
FOGIN
GINFO
INFOG
NFOGI
OGINF
```

Se cunosc caracterele de pe ultima coloană a matricei B și se cere determinarea caracterelor de pe prima linie a acesteia.

Date de intrare

Fișierul de intrare **LITERE.IN** conține o singură linie pe care se află caracterele de pe ultima coloană a matricei B.

Date de ieșire

Fișierul de ieșire **LITERE.OUT** va conține o singură linie pe care se vor afla caracterele de pe prima linie a matricei B.

Restricții și precizări

- șirul inițial conține cel mult 10000 de caractere;
- pe baza ultimei coloane se va putea determina întotdeauna prima linie.

Exemplu

LITERE.IN	LITERE.OUT
NOGIF	FOGIN

Timp maxim de execuție/test: 1 secundă

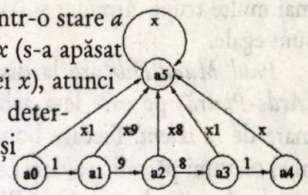
P060223: Zăvor

O închizătoare cu cifru este un instrument cu zece butoane pentru tastarea a câte unei cifre și un buton de reset care folosește la aducerea închizătorii în starea inițială.

Închizătoarea mai dispune de un buton cu care aceasta poate fi deschisă, cu condiția să fi tastat în prealabil parola corectă.

În închizătoare se încorporează un cip pentru o parolă dată, care funcționează ca un automat finit (rezultă că dacă parola este formată din N cifre, atunci automatul are exact $N + 2$ stări).

Dacă automatul se află într-o stare a și recepționează un semnal x (s-a apăsat butonul corespunzător cifrei x), atunci ajunge într-o stare nouă b, determinată în mod univoc de a și x.



Automat pentru cheia 1981

De exemplu, dacă parola este 1981, atunci un automat care funcționează corect, trebuie să-și schimbe stările conform figurii alăturate. De asemenea, automatul are voie să deschidă închizătoarea doar în starea $a4$. Pe figură xi reprezintă toate cifrele diferite de i, iar x reprezintă toate cifrele.

Verificați dacă cipul dat funcționează corect sau nu. Se va verifica doar faptul că pe baza semnalelor se va deschide sau nu închizătoarea; dacă se recepționează parola corectă, închizătoarea se deschide întotdeauna.

Pentru această testare aveți la dispoziție trei operații; pentru programatorii în *Pascal*, acestea sunt conținute în *unit*-ul **ZAVOR.TPU**:

- **procedure** ResetA: automatul este adus în starea de bază;
- **procedure** Schimba(x:Char): automatul trece din starea curentă în starea corespunzătoare semnalului x ('0' ≤ x ≤ '9');
- **function** Deschide:Char: dacă starea curentă este cea de deschidere, atunci valoarea returnată este 'D', altfel se returnează valoarea 'N'.

Programatorii în *C/C++* vor avea la dispoziție fișierul *header* **ZAVOR.H** care va conține funcțiile:

- **void** ResetA(**void**): automatul este adus în starea de bază;
- **void** Schimba(**char** x): automatul trece din starea curentă în starea corespunzătoare semnalului x ('0' ≤ x ≤ '9');
- **char** Deschide(**void**): dacă starea curentă este cea de deschidere, atunci valoarea returnată este 'D', altfel se returnează valoarea 'N'.

Date de intrare

Singura linie a fișierului de intrare **TEST.IN** conține parola care este formată din cel mult 100 cifre zecimale (caractere cuprinse între '0' și '9') nedespărțite prin spații. Ultimul caracter este marcajul de sfârșit de linie.

Date de ieșire

În fișierul de ieșire **TEST.OUT** se va scrie rezultatul pe o singură linie.

Dacă cifrul se deschide doar pentru parola dată în fișierul de intrare, atunci în fișier se scrie cuvântul CORECT. Dacă cifrul se deschide și pentru alte chei decât parola dată, în fișierul de ieșire se va scrie un șir de caractere care deschid cifrul.

Timp maxim de execuție/test: 1 secundă

CONCURS de programare

Ediția a patra a concursului de programare organizat de Gazeta de Informatică începe pe data de 14 octombrie 2002. Pentru detalii referitoare la acest concurs citiți paginile următoare sau vizitați secțiunea de pe site-ul nostru dedicată concursului: www.ginfo.ro/concurs.

Informații generale

Concursul *Bursele Agora* este o competiție internațională destinată elevilor și studenților. Acest concurs este organizat de redacția *GInfo* și editura *Agora Media*.

Vor fi douăzeci de runde, la fiecare dintre ele participanții trebuie să rezolve câte o problemă și să trimită soluțiile prin *e-mail*; adresa *e-mail* a concursului nostru este **concurs@ginfo.ro**. După cele douăzeci de etape, care se vor desfășura prin *e-mail*, va fi o etapă finală care va avea loc la *Cluj-Napoca*; doar primii zece clasați în urma celor douăzeci de runde vor participa la faza finală.

Problemele care trebuie rezolvate sunt propuse de redacția *GInfo* în colaborare cu studenți și profesori care se ocupă de pregătirea elevilor români care participă la diferite concursuri naționale și internaționale.

Înscrierea

Pentru a participa la această ediție a concursului de programare *Bursele Agora* va trebui, mai întâi, să vă înscrieți. Cea mai simplă modalitate de înscriere este completarea *online* a formularului de înscriere (**www.ginfo.ro/concurs/inscriere.shtml**) care va fi trimis automat, prin intermediul clientului dumneavoastră de *e-mail*, la **concurs@ginfo.ro**.

O altă posibilitate o constituie trimiterea unui *e-mail* la adresa **concurs@ginfo.ro**. Acesta va trebui să conțină informațiile cerute în formularul disponibil *online* și anume: numele și prenumele dumneavoastră, data nașterii, adresa *e-mail*, adresa poștală și numărul de telefon.

Cea de-a treia posibilitate constă în trimiterea unei scrisori care va conține aceleași informații. Aceasta trebuie trimisă pe adresa:

Redacția GInfo
str. Universității nr. 8/8, 3400 Cluj-Napoca

În termen de cinci zile de la primirea de către redacția *GInfo* a *e-mail*-ului sau a scrisorii dumneavoastră, vă vom trimite un *e-mail* care va conține codul dumneavoastră de identificare pentru ediția 2002/2003 a concursului.

Desfășurare concursului

Limbajele de programare acceptate la concursul *Bursele Agora* sunt *Pascal* și *C/C++*. Fișierele sursă vor fi compilate folosind *Borland Pascal* 7.0 sau *Borland C++* 3.1.

Problemele vor fi publicate pe *site*-ul *Web* al revistei noastre într-o secțiune dedicată acestui scop (www.ginfo.ro/concurs/probleme.shtml). Rezolvările vor trebui trimise înaintea expirării termenului limită specificat pentru fiecare rundă.

O singură soluție va fi acceptată pentru fiecare problemă. Dacă un concurent va trimite mai multe soluții pentru o anumită problemă, numai prima va fi luată în considerare, restul fiind ignorate. Redacția *GInfo* își rezervă dreptul de a descalifica concurenții care trimit soluții identice (sau aproape identice).

Este interzisă folosirea de secvențe scrise în limbaj de asamblare; dacă un fișier sursă conține astfel de secvențe concurentul va primi 0 puncte pentru problema de la runda respectivă.

Este interzis ca programele generate folosind sursele concurenților să modifice *timer*-ul. Concurenții care vor încerca să modifice ceasul sistem vor fi descalificați.

Dacă aveți neclarități referitoare la enunțurile problemelor, puteți adresa întrebări folosind adresa *e-mail* a concursului. Nu puteți adresa decât întrebări ale căror răspunsuri posibile sunt `DA` sau `NU`. Dacă răspunsul poate fi găsit în enunțul problemei sau întrebarea nu respectă condiția anterioară, atunci răspunsul la întrebare va fi `FĂRĂ COMENTARII`. În cazul în care redacția *GInfo* consideră că răspunsul la o anumită întrebare trebuie primit de toți participanții, atunci cel care a adresat întrebarea va primi răspunsul `ANUNȚ GENERAL`. Anunțurile generale vor fi disponibile *online* în secțiunea dedicată problemelor.

Fiecare etapă se va desfășura pe parcursul unei săptămâni. Enunțul va fi disponibil *online* în ziua de ***luni*** a săptămânii, pot fi adresate întrebări până ***marți***, toate anunțurile generale vor fi disponibile în cursul zilei de ***miercuri***, soluțiile trebuie trimise până ***vineri***, vor fi corectate ***sâmbăta***, iar rezultatele vor fi publicate în cursul ***duminicii***.

Trimiterea soluțiilor

Pentru a trimite soluțiile de la o anumită rundă va trebui să trimiteți un *e-mail* având ca subiect codul dumneavoastră de identificare urmat de textul `Runda XX`, unde `XX` reprezintă numărul rundei (`01-20`). Acest *e-mail* va avea atașată o arhivă `.ZIP`, `.ACE` sau `.RAR` numită **`YYYYYRXX.EXT`** unde

YYYYY este codul dumnevoastră de identificare, XX este numărul rundei, iar EXT este extensia (RAR, ACE sau ZIP).

Arhiva va conține două fișiere. Primul dintre ele va fi un fișier .BAT care va conține linia de comandă pe care doriți să o folosim pentru a compila fișierul sursă (al doilea fișier al arhivei), corespunzător rundei respective. Acest prim fișier al arhivei se va numi **YYYYYRXX.BAT** unde YYYYY este codul dumneavoastră de identificare, iar XX este numărul rundei. Cel de-al doilea fișier din arhivă va conține codul sursă corespunzător rezolvării pe care doriți să o trimiteți pentru problema de la runda respectivă. Denumirea acestuia va fi **YYYYYRXX.PAS**, **YYYYYRXX.C** sau **YYYYYRXX.CPP**. Din motive de securitate, arhivele autoexpandabile nu sunt acceptate.

Arhiva nu va conține nici un alt fișier sau director, deci nu puteți trimite fișiere auxiliare împreună cu fișierele sursă. Dacă programul dumneavoastră trebuie să creeze fișiere auxiliare, atunci acestea vor avea extensia .TMP, vor fi create în directorul curent, și trebuie șterse înaintea terminării execuției programului. Dacă această regulă nu este respectată concurentul va primi 0 puncte pentru problema corespunzătoare, chiar dacă soluția este corectă. Dacă redacția *GInfo* consideră că un concurent pune în pericol buna desfășurare a competiției (prin suprascrierea unor fișiere, crearea de fișiere foarte mari etc.), atunci acel concurent poate fi descalificat.

Toate fișierele trimise devin proprietatea editurii *Agora Media* și nu pot fi folosite în scopuri comerciale fără acordul scris al editurii.

Evaluarea

Pentru fiecare problemă puteți primi maximum 100 de puncte; numărul maxim de puncte pentru întreg concursul este 2000. Pentru corectarea fiecărei probleme vom folosi între 5 și 20 de seturi de date; dacă programul va genera un fișier de ieșire corect pentru un anumit set de date, atunci concurentul va primi un număr de puncte care se va adăuga la numărul de puncte corespunzător problemei respective.

Evaluarea va fi automată, în vederea efectuării acesteia se va folosi întotdeauna ultima versiune a programului de evaluare *GInfo Evaluator*.

Calculatorul folosit pentru evaluare are următoarea configurație: procesor *AMD Athlon XP* 1600+ (1,4 GHz), RAM: 128 MB, HDD: 13 GB.

Este garantat faptul că în timpul evaluării vor fi liberi cel puțin 400 KB de memorie convențională și cel puțin 100 MB de spațiu pe disc. Sistemul de operare folosit în timpul evaluării va fi *Microsoft Windows XP Professional*. Dacă un program va efectua o operație ilegală, atunci concurentul va primi 0 puncte pentru setul de date corespunzător.

În cazul în care configurația *hardware* sau *software* a calculatorului folosit pentru evaluare se modifică, un anunț în acest sens va fi disponibil pe pagina principală a *site*-ului nostru (**www.ginfo.ro**).

Sfaturi

Programatorii în *Pascal* nu ar trebui să folosească niciodată *unit*-ul CRT, deoarece programele lor se vor opri returnând eroarea de execuție Error 200: Division by zero. Acest fapt este cauzat de un bine cunoscut *bug* al mediului de programare *Borland Pascal 7.0*; nici o problemă propusă spre rezolvare în cadrul concursului *Bursele Agora* nu necesită folosirea acestui unit.

Atunci când sunt folosite anumite tipuri de date (cum ar fi Comp) modul 8087 trebuie să fie activ. Opțiunile de mediu permit setarea acestui mod ca mod implicit, dar în mod normal acest mod nu este activ. Programatorii în *Pascal* sunt rugați să includă în codul sursă directiva de compilare {$N+} dacă doresc să folosească tipuri de date care necesită activarea acestui mod. În caz contrar, în mometul în care vom încerca să compilăm fișierul sursă, va fi generată eroarea de compilare Error 116: Must be in 8087 mode to compile this. Pentru a vă asigura că programul se va compila exact ca pe calculatorul folosit pentru a-l scrie, apăsați CTRL+O,O. Astfel, în sursa dumneavoastră, vor fi incluse toate directivele de compilare cu opțiunile care sunt setate în mediu. Puteți modifica aceste opțiuni astfel încât să fie cele dorite de dumneavoastră. Nu vă îngrijorați dacă nu cunoașteți semnificația unor anumite directive de compilare; acestea nu vor avea efecte negative asupra execuției programului - dacă programul dumneavoastră va rula corect pe calculatorul pe care l-ați folosit pentru a-l scrie, atunci va rula corect și pe calculatorul folosit pentru evaluare.

Toate programele trebuie să citească datele de intrare din fișierul de intrare specificat. Nu ar trebui să fie așteptate date de la intrarea standard (tastatură). Dacă programul dumneavoastră va aștepta o reacție de la tastatură, atunci execuția sa va fi oprită după expirarea timpului și nu veți obține nici un punct pentru acea soluție chiar dacă ea este corectă. Deci, vă rugăm să nu includeți în codul sursă secvențe de tipul Readln, getch(), getche() etc.

Toate programele trebuie să scrie datele de ieșire în fișierul de ieșire specificat. Puteți scrie date și la ieșirea standard, dar aceasta nu va duce decât la încetinirea programului, deci nu este recomandabil să scrieți date pe ecran. Toate datele scrise la ieșirea standard sunt ignorate în timpul evaluării. Programul de evaluare va verifica numai soluția scrisă în fișierul de ieșire specificat. Fișierul de intrare se află întotdeauna în directorul curent; nu încercați să citiți date din alte locații. Fișierul de ieșire va trebui generat tot în directorul curent; nu scrieți date de ieșire în alte locații. Trebuie să folosiți întotdeauna denumirile specificate în enunț pentru fișierul sursă, fișierul de intrare și fișierul de ieșire. Folosirea altor denumiri va duce la funcționarea incorectă a programului.

Exemplu

Denumirea fișierului de intrare: **SUMA.IN**
Denumirea fișierului de ieșire: **SUMA.OUT**
Descriere: Dându-se două numere, aflați suma lor.

Date de intrare: Fișierul de intrare conține, pe o singură linie, cele două numere separate printr-un singur spațiu.
Date de ieșire: Fișierul de ieșire ar trebui să conțină suma celor două numere.
Exemplu:

SUMA.IN	**SUMA.OUT**
2 4	6

Timp maxim de execuție: 1 secundă/test

După citirea enunțului veți începe să rezolvați problema. Veți observa că nu este clar dacă numerele sunt întregi sau reale.

Pentru a clarifica acest lucru va trebui să adresați o întrebare. Aceasta nu poate fi: *Numerele sunt întregi sau reale?* deoarece nu se poate răspunde la ea prin DA sau NU. O soluție ar fi să întrebați: *Numerele sunt întregi?* Veți primi răspunsul NU și veți ști că numerele sunt reale.

Acum apare o întrebare mult mai importantă: *Ce precizie trebuie folosită pentru afișarea rezultatului?* Se pare că redacția *GInfo* a uitat să precizeze acest lucru în enunț. Bineînțeles, va trebui să adresați o întrebare la care să se poată răspunde prin DA sau NU. Ați putea încerca: *Este suficientă o precizie de trei zecimale?* Răspunsul la întrebare va fi ANUNȚ GENERAL și va fi urmat de publicarea în secțiunea dedicată problemelor (**www.ginfo.ro/concurs/probleme.shtml**) a unui mesaj de genul:

Rezultatul va trebui scris cu o precizie de două zecimale. Ne cerem scuze pentru descrierea incompletă a problemei.

Sperăm să nu fim nevoiți să publicăm prea des astfel de mesaje, dar... După citirea anunțului general puteți începe scrierea codului sursă. O soluție corectă ar putea fi:

```
 1: Program SUMA;
 2: var a,b:Real;
 3:     f:Text;
 4: Begin
 5:   Assign(f,'SUMA.IN');
 6:   Reset(f);
 7:   Readln(f,a,b);
 8:   Close(f);
 9:   Assign(f,'SUMA.OUT');
10:   Rewrite(f);
11:   Writeln(f,a+b:0:2);
12:   Close(f)
13: End.
```

Numerele de la începutul fiecărei linii nu fac parte din program; ele sunt folosite pentru a ne putea referi mai ușor la o anumită linie.

Dacă veți adăuga o linie `1': uses Crt;` și o linie `4': Clrscr;` nu veți reuși să obțineți decât eroarea `Error 200: Division by zero` așa cum am explicat în secțiunea *Sfaturi*.

Dacă adăugați o linie `12': Readln;` rezultatul evaluării va fi `Time out!`, deoarece execuția programului nu se va termina, ci se va aștepta apăsarea tastei <**ENTER**>. Execuția va fi întreruptă de către programul de evaluare după o secundă.

Dacă folosiți linia `5: Assign(f,'C:\SUMA.IN');` atunci programul va funcționa corect numai dacă directorul curent este rădăcina *drive*-ului `C:` (ceea ce este foarte improbabil). Nu veți obține decât eroarea `File not found`.

Dacă folosiți linia `5: Assign(f,'ALTFISIER');` atunci programul nu va citi datele de intrare din fișierul specificat și va rula incorect.

Dacă folosiți linia `9: Assign(f,'C:\SUMA.OUT');` atunci programul va părea că funcționează corect, dar programul de evaluare nu va găsi fișierul dumneavoastră de ieșire decât dacă directorul curent este rădăcina *drive*-ului `C:` (din nou, șansele sunt foarte mici). Programul de evaluare va returna `No output file!`.

Dacă folosiți linia `9: Assign(f,'ALTFISIER');` atunci programul nu va scrie datele în fișierul de ieșire specificat, iar programul de evaluare va returna `No output file!`.

Dacă folosiți linia `7: Readln(a,b);` atunci programul va aștepta datele de intrare de la intrarea standard și va fi întrerupt după o secundă. Puteți folosi această linie numai dacă aveți o secvență asemănătoare cu:

```
5: Assign(Input,'SUMA.IN');
6: Reset(Input);
7: Readln(a,b);
8: Close(Input);
```

Dacă folosiți linia `11: Writeln(a+b:0:2);` atunci datele de ieșire vor fi scrise pe ecran și nu în fișierul de ieșire specificat. Puteți folosi această linie numai dacă secvența este de tipul:

```
 9: Assign(Output,'SUMA.OUT');
10: Rewrite(Output);
11: Writeln(a+b:0:2);
12: Close(Output)
```

Acestea sunt cele mai des întâlnite erori care au apărut la precedentele ediții ale concursului organizat de *GInfo*. Sperăm că nu veți face aceleași greșeli și la această ediție.

După scrierea codului sursă trebuie să trimiteți soluția. În cele ce urmează vom considera că *ID*-ul dumneavoastră este `02457` și participați la prima rundă a concursului . Trebuie să creați arhiva pe care o veți trimite; aceasta se va numi **02457R01.ZIP**, **02457R01.ACE** sau **02457R01.RAR** și va conține două fișiere: **02457R01.BAT** și **02457R01.PAS**. Fișierul **02457R01.PAS** va conține codul sursă al rezolvării, în timp ce fișierul **02457R01.BAT** va conține linia de comandă care va fi folosită pentru compilare, de exemplu `bpc 02457R01.PAS`. Folosiți `bpc` pentru compilarea surselor *Pascal* și `bcc` pentru compilarea surselor *C/C++*.

După crearea acestui fișier trebuie construită arhiva pe care o veți trimite. Folosiți un arhivator care creează arhive `.RAR`, `.ACE` sau `.ZIP`; nu creați o arhivă autoexpandabilă. Acum puteți trimite *e-mail*-ul care va conține soluția. Nu uitați să atașați arhiva și să scrieți în linia *Subject*: `02457 Runda 01`.

RECUNOAȘTEREA formelor

Radu-Daniel Vatavu

În cadrul acestui serial vom prezenta mai multe detalii referitoare la modul în care pot fi detectate asemănările dintre obiectele grafice. Acest prim episod va conține prezentarea unor noțiuni introductive, precum și un algoritm simplu de clasificare a formelor.

Introducere

Recunoașterea formelor este un domeniu al inteligenței artificiale, care cuprinde ansamblul tehnicilor de reprezentare și decizie în vederea automatizării procesului de identificare a asemănărilor dintre obiecte sau fenomene. Metodele sale își găsesc aplicații într-o varietate de domenii, dintre care amintim:

- *medicină* (sisteme automate de diagnosticare pe baza simptomelor prezentate și a rezultatelor analizelor, analiza și interpretarea electro-cardiogramelor și encefalogramelor, analiza radiografiilor);
- *industrie* (detectarea defectelor de fabricație ale produselor, sisteme de viziune artificială pentru roboți industriali);
- *procesarea semnalelor și imaginilor* (recunoașterea vorbirii, recunoașterea caracterelor, a amprentelor, recunoașterea facială, analiza imaginilor satelit);
- *analiza diferitelor fenomene economice și luarea unor decizii de natură financiară, analiza fenomenelor sociale*, etc.

O aplicație standard de recunoștere a formelor cuprinde următoarele etape:

- ***codificarea***: folosirea unei metode de reprezentare a datelor de prelucrat compatibilă cu cerințele calculatorului numeric. Această etapă este realizată în anumite cazuri de un translator, care poate fi o cameră video într-o aplicație de recunoaștere facială sau un microfon în cazul recunoașterii vorbirii;
- ***preprocesarea***: eliminarea eventualelor zgomote care pot afecta datele de intrare, efectuarea de prelucrări în vederea reducerii dimensiunilor datelor, normalizări, transformări liniare și neliniare etc;
- ***învățarea***: construirea unui model de clasificator, plecând de la o mulțime de exemple și împărțirea acestora într-un anumit număr de clase;
- ***analiza și decizia***: stabilirea clasei din care face parte o formă, inițial necunoscută;
- ***postprocesarea***: eventuala validare a deciziei luate, având ca suport o anumită bază de cunoștințe.

Din aceste etape se desprinde obiectul general al recunoașterii formelor: pe de o parte descrierea și reprezentarea obiectelor și fenomenelor într-o manieră standard de "formă" astfel încât să poată fi supuse unor prelucrări matematice și algoritmice și pe de altă parte, luarea unei decizii privind clasificarea unei forme necunoscute la un ansamblu deja "învățat". Evident, de o mare importanță se dovedește a fi alegerea reprezentării, care trebuie să conțină suficiente informații pentru a asigura o precizie ridicată clasificării, dar trebuie să fie și condensată pentru a evita un timp de calcul exagerat în momentul luării deciziei.

Există două mari abordări în domeniul recunoașterii formelor, în funcție de manierele diferite de reprezentare, învățare și luare a deciziilor, și anume:

- recunoașterea numerică sau statistică;
- recunoașterea sintactică sau structurală.

În cele ce urmează, vor fi abordate aspectele referitoare la recunoașterea statistică a formelor.

Noțiuni generale

Abordarea statistică are în vedere clasificarea obiectelor sau fenomenelor, având la bază masurarea unor anumite proprietăți sau ***caracteristici*** ale acestora. Astfel, o *formă* poate fi reprezentată ca un *ansamblu de caracteristici.*

De exemplu, într-o aplicație de recunoaștere a caracterelor pot fi utilizate drept caracteristici aria, perimetrul, înălțimea, lățimea, raportul dintre aria caracterului și aria dreptunghiului circumscris etc. La fel de bine, împărțind dreptunghiul circumscris caracterului în $m \times n$ dreptunghiuri, pot fi alese drept caracteristici factorii de umplere ai fiecărui dreptunghi (definiți ca raportul dintre numărul de pixeli aparținând caracterului aflați în interiorul dreptunghiului k și aria în pixeli a dreptunghiului k, $k = 0 .. m \times n-1$). Cele două modalități de alegere a caracteristicilor sunt prezentate în figura 1.

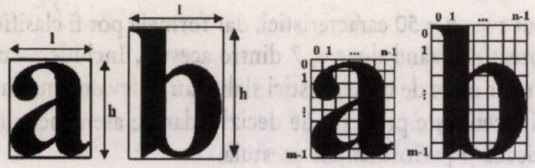


Figura 1

În continuare, prin *formă* se va înțelege modelarea matematică a obiectelor sau fenomenelor sub forma unor vectori de caracteristici. Astfel, fiecărei forme x i se asociază un punct din spațiul R^p (numit și spațiul caracteristicilor):

$$x = (x_0, x_1, \ldots, x_{p-1})$$

Prin *spațiul formelor* se înțelege populația formelor care urmează a fi analizate, deci o submulțime F a spațiului R^p. *Spațiul claselor* (sau mulțimea C a claselor) desemnează categoriile care servesc la clasificarea formelor date. În exemplul considerat, spațiul formelor poate fi submulțimea F a matricelor de ordin $m \times n$:

$$F \subset \{A_{n\times m} \mid a_{ij} \in [0,1]; i = 0..m-1; j = 0..n-1\},$$

iar spațiul claselor mulțimea $C = \{a, b, c, \ldots, z\}$. Prin *clasificator* se înțelege o funcție K definită pe spațiul formelor ce asociază fiecărei forme x din F un identificator aparținând spațiului claselor:

$$K : F \to C, K(x) = c$$

Se numește *set de învățare* o mulțime de forme a căror clasificare este cunoscută:

$$S = \{(x,c)/x \in F, c \in C\}$$

Metoda de învățare utilizată va influența performanțele și precizia clasificării. Se pot distinge două modalități generale de învățare:

- *învățarea supravegheată*: presupune construirea clasificatorului general pe baza unui set de învățare S (se cunoaște deci clasa de apartenență a fiecărei forme);
- *învățarea nesupravegheată*: presupune realizarea unei partiționări a spațiului formelor, ținând cont de "asemănările" formelor furnizate (sunt construite grupuri de forme asemănătoare, fără nici o informație inițială privind clasele de apartenență ale acestora).

Clasificarea caracteristicilor

Caracteristicile utilizate pentru reprezentarea formelor pot fi clasificate în funcție de natura lor în:

- caracteristici *cantitative*;
- caracteristici *calitative*.

Caracteristici cantitative

Fie o submulțime S' a setului de învățare S, definit mai sus, astfel încât toate formele să prezinte o clasificare în aceeași clasă. De asemenea, presupunem că pentru descrierea formelor au fost alese n caracteristici reprezentative:

$$S' = \{x_0, x_1, \ldots x_{p-1}\} \quad K(x_i) = c \in C$$
$$x_i = (x_{i,0}, x_{i,1} \ldots x_{i,n-1}) \quad i = 0..p-1$$

Etapa de alegere a caracteristicilor este esențială, influențând rezultatele viitoare ale clasificatorului și deci, ale sistemului de recunoaștere. Drept consecință, se impune o analiză a valorilor care pot fi luate de aceste caracteristici. Analiza caracteristicilor cantitative se realizează cu ajutorul unei serii de indicatori statistici: indicatori medii și indicatori ai variației prezentați în tabelele 1 și 2.

Denumire	Modalitate de calcul
media aritmetică	pentru o caracteristică j, este dată de formula: $m_j = \frac{\sum_{k=0}^{p-1} x_{k,j}}{p}$
modulul	reprezintă valoarea caracteristicii cu cea mai mare frecvență de apariție;
mediana	reprezintă acea valoare a caracteristicii situată în mijlocul șirului care conține valorile înregistrate ale acesteia, aranjate în ordine crescătoare sau descrescătoare; valoarea medianei pentru caracteristica j este: $Me_j = \begin{cases} x_{[p/2]} & \text{pt. } p \text{ impar} \\ \frac{x_{[p-1/2]} + x_{[p/2]}}{2} & \text{pt. } p \text{ par} \end{cases}$

Tabelul 1: Indicatori medii utilizați în analiza caracteristicilor

Un indicator al corelației dintre caracteristici este reprezentat de covarianță. *Covarianța* dintre două caracteristici i și j este definită ca media aritmetică a produselor dintre abaterile valorilor fiecărei caracteristici față de valorile medii:

$$\text{cov}(i,j) = \frac{\sum_{k=0}^{p-1} (x_{k,i} - m_i) \bullet (x_{k,j} - m_j)}{p}$$

Semnificația indicatorului este prezentată în continuare:

- $\text{cov}(i,j) > 0$: legătura dintre cele două caracteristici este directă (cresc sau descresc împreună);
- $\text{cov}(i,j) < 0$: legătura dintre caracteristici este de sens invers;
- $\text{cov}(i,j) = 0$: caracteristicile sunt independente.

Trebuie remarcată existența relației:

$$\text{cov}(i,j) = \sigma^2_i$$

Caracteristici calitative

Uneori, caracteristicile cantitative nu sunt suficiente pentru a caracteriza corespunzător un obiect sau un fenomen, există dificultăți privind măsurarea lor sau pur și simplu există caracteristici calitative mult mai relevante. Un exemplu de caracteristică calitativă poate fi: *forma* = {*alungită*, *rotundă*, *ovală*} în descrierea unor obiecte sau *caracterul* = {*flegmatic*, *melancolic*, *sangvinic*, *coleric*} în descrierea unor persoane. Se observă că setul valorilor posibile pentru caracteristicile calitative este finit. De asemenea, în vederea prelucrării lor sub o formă matematică, se impune un sistem de codificare a acestor caracteristici.

Denumire	Modalitate de calcul
amplitudinea variației	diferența dintre nivelul maxim și nivelul minim al caracteristicii: $A_j = \max(x_{k,j}) - \min(x_{k,j}),\ k = 0..p-1$
abaterea medie liniară	pentru caracteristica j se calculează ca media aritmetică simplă a abaterilor valorilor caracteristicilor față de valoarea medie: $d_j = \frac{\sum_{k=0}^{p-1} \lvert x_{k,j} - m_j \rvert}{p}$
abaterea medie pătratică	pentru caracteristica j se calculează ca media pătratică a abaterilor valorilor caracteristicilor față de valoarea medie: $\sigma_j = \sqrt{\frac{\sum_{k=0}^{p-1} (x_{k,j} - m_j)^2}{p}}$
coeficientul de variație	pentru caracteristica j se determină ca raportul dintre abaterea medie pătratică și media aritmetică a caracteristicii: $v_j = \frac{\sigma_j}{m_j}$ *coeficientul de variație are o importanță deosebită, furnizând informații despre gradul de omogenitate al valorilor caracteristicii; dacă depășește valoarea de 35-40% (exprimat procentual), atunci media valorilor caracteristicii j nu este reprezentativă;*
dispersia	media aritmetică a abaterilor valorilor caracteristicii față de valoarea medie: $\sigma_j^2 = \frac{\sum_{k=0}^{p-1} (x_{k,j} - m_j)^2}{p}$

Tabelul 2: Indicatori ai variației utilizați în analiza caracteristicilor

Alegerea și selecția caracteristicilor

Etapa de alegere a caracteristicilor care vor descrie formele este de o importanță majoră, influențând atât precizia și calitatea clasificatorului cât și timpul de execuție al algoritmului de clasificare. Ansamblul caracteristicilor alese trebuie deci să se supună unor cerințe de ordin calitativ și cantitativ.

Presupunem în continuare că a fost ales un set de n caracteristici care vor descrie un ansamblu de obiecte sau fenomene. Adesea, multe din caracteristici au relativ puțină importanță în discriminarea dintre clase. De exemplu, putem avea $n = 50$ caracteristici, dar formele pot fi clasificate corect utilizând doar 6, 7 dintre acestea. Includerea unui număr mare de caracteristici slabe sau irelevante nu numai că încetinește procesul de decizie, dar de asemenea diminuează și performanțele acestuia.

În această privință există o serie de abordări euristice care conduc la eliminarea acelor caracteristici cu importanță relativ redusă, și anume:

- utilizarea unor algoritmi bazați pe forță brută (de tipul *backtracking*) care vor alege subsetul de $m < n$ caracteristici care conferă cea mai mică rată de eroare a clasificării; abordarea prezintă anumite dezavantaje:
  - metoda este viabilă doar pentru valori mici ale numărului n de caracteristici inițiale;
  - există posibilitatea obținerii unui set de m caracteristici adaptate perfect pentru setul de învățare, dar cu o insuficientă capacitate de generalizare;
- utilizarea unor algoritmi de tip *greedy*; există două variante care pot fi aplicate și anume: pornind de la o submulțime alcătuită dintr-o singură caracteristică, se adaugă progresiv o nouă caracteristică ce îmbunătățește performanța clasificării sau, pornind de la mulțimea inițială, se elimină caracteristica ce reduce cel mai puțin performanța totală;
- combinarea caracteristicilor; această operație conduce la obținerea unor caracteristici noi pe baza combinării celor existente, care pot determina performanțe sporite ale clasificării; o metodă simplă și uzuală este reprezentată de transformarea liniară a caracteristicilor, noul subset fiind dat de relația:

$y^T = A \cdot x^T$, unde

$y = (y_0, y_1, \ldots y_{m-1})$, y^T – vectorul y transpus

$x = (x_0, x_1, \ldots x_{n-1})$, x^T – vectorul x transpus

A – matrice de ordin $m \times n$

O altă soluție constă în utilizarea matricei de covarianță definită astfel:

$$C = (c_{i,j})_{n \times n},\ c_{i,j} = \mathrm{cov}(i, j)$$

Se determină valorile proprii și vectorii proprii ai matricei C și se construiește matricea de transformare A din vectorii proprii corespunzători celor mai mari m valori proprii;

- analiza corelației dintre caracteristici; această abordare este utilă în cazul în care pot exista caracteristici corelate (de exemplu, între caracteristicile i și j poate exista o dependență de tipul $x_i = ax_j + b$ etc.). Metodele de analiză a corelației dintre carateristici se regăsesc în metodele generale utilizate în statistică, dintre care amintim:
  - metode parametrice sau analitice (calcularea raportului de corelație, coeficientul de corelație, etc.);
  - metode neparametrice (coeficientul de asociere, coeficienți de corelație a rangurilor *Spearman* și *Kendall*, coeficientul *Pearson*, raportul de corelație empirică, etc.).

Metodele prezentate nu asigură întotdeauna obținerea celui mai bun subset de caracteristici pentru un anumit proces de reprezentare și clasificare, însă pot contribui la creșterea performanțelor acestuia și au avantajul de a fi ușor de implementat. Mai mult, se pot folosi combinații ale acestor metode în vederea obținerii unor rezultate mai bune (exemplul clasic de aplicare al unor algoritmi de tip *greedy* pentru reducerea dimensiunilor spațiului soluțiilor unui algoritm de tip *backtracking*).

Distanțe

Fie F mulțimea formelor definită mai sus. Se numește ***distanță*** o funcție:

$$d : F \times F \to \Re$$

care respectă următoarele axiome:

$d(x_1, x_2) \geq 0 \quad \forall\, x_1, x_2 \in F$

$d(x_1, x_2) = 0 \Leftrightarrow x_1 = x_2$

$d(x_1, x_2) = d(x_2, x_1) \quad \forall\, x_1, x_2 \in F$

$d(x_1, x_2) \leq d(x_1, x_3) + d(x_3, x_2) \quad \forall\, x_1, x_2, x_3 \in F$

În tabelul 3 sunt prezentate câteva exemple de distanțe des utilizate.

Denumire	Modalitate de calcul
distanța euclidiană	$d(x_1,x_2) = \sqrt{\sum_{j=0}^{n-1} (x_{1,j} - x_{2,j})^2}$
distanța Manhatten	$d(x_1,x_2) = \sum_{j=0}^{n-1} \lvert x_{1,j} - x_{2,j} \rvert$
distanța Cebâșev	$d(x_1,x_2) = \max \lvert x_{1,j} - x_{2,j} \rvert, j = 0..n-1$
distanța Mahalanobis	$d(x_1,x_2) = \sqrt[n]{\sum_{j=0}^{n-1} (x_{1,j} - x_{2,j})^n}$

Tabelul 3: Exemple de distanțe

În figura 2 este propusă o reprezentare intuitivă a celor patru distanțe pentru cazul particular al planului ($n = 2$ caracteristici), punându-se în evidență modul în care se modifică locul geometric al punctelor din plan (formelor) egal depărtate de un punct fix (formă prototip), în funcție de tipul de distanță folosită.

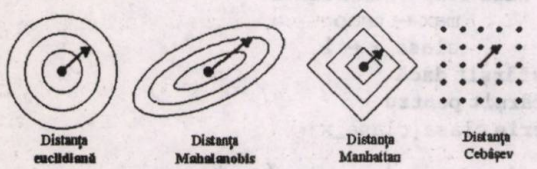


Figura 2

Tipurile de distanțe prezentate în tabelul 3 au la bază modelul spațiului cu n dimensiuni R^n. Aplicarea lor pentru determinarea distanțelor dintre două forme trebuie să țină seama de diferențele dintre caracteristici și de faptul că acestea nu prezintă aceeași unitate de măsură. Calcularea distanței dintre două forme x_1 și x_2 este corectă doar în cazul în care toate caracteristicile prezintă valori care se întind de-a lungul aceleiași scale. Ca urmare, se impune o scalare a valorilor caracteristicilor iar relația pentru calculul distanței euclidiene dintre formele x_1 și x_2 devine:

$$d(x_1, x_2) = \sqrt{\frac{\sum_{j=0}^{n-1} \left(\frac{x_{1,j} - x_{2,j}}{\sigma_j} \right)^2}{n}}$$

unde σ_j reprezintă abaterea medie pătratică a caracteristicii j prezentată în tabelul 2. Celelalte formule sunt modificate corespunzător.

Similarități

Fie F mulțimea formelor. Se numește **indice de similaritate** funcția $s : F \times F \to \Re$ care respectă următoarele axiome:

$s(x_1, x_2) = s(x_2, x_1) \quad \forall\, x_1, x_2 \in F$

$s(x_1, x_1) = s(x_2, x_2) \geq s(x_1, x_2) \quad \forall\, x_1, x_2 \in F$

Un exemplu de indice de similaritate este:

$$s(x_1, x_2) = d_{max} - d(x_1, x_2) \quad \forall\, x_1, x_2 \in F$$

unde d este o distanță oarecare, iar

$$d_{max} = \max\{d(x_1, x_2) / x_1, x_2 \in F\}$$

Un caz particular pentru calculul indicilor de similaritate îl reprezintă formele cu caracteristici binare care indică prezența sau absența unui anumit atribut. Astfel, considerând două forme x_1 și x_2 reprezentate prin intermediul a n caracteristici binare, se consideră următorii indici:

- a = numărul de atribute deținute de ambele forme
- b = numărul de atribute deținute numai de forma x_1
- c = numărul de atribute deținute numai de forma x_2
- d = numărul de atribute care nu sunt deținute de nici una din cele două forme
- $p = a + b + c + d$

Pentru exemplificare, să considerăm un ansamblu de patru persoane (deci patru forme) descrise prin intermediul a două atribute (caracteristici): *caracterul* = {*flegmatic*, *melancolic*, *sangvinic*, *coleric*} și *vârsta* = {*tânăr*, *matur*, *bătrân*}.

Valorile caracteristicilor sunt prezentate în tabelul 4. Pe baza indicilor a, b, c, d, p definiți anterior, se pot calcula o serie de indici de similaritate s, care sunt prezentați în tabelul 5.

Un algoritm simplu de clasificare

În construirea unui clasificator care va fi utilizat într-o aplicație specifică de recunoaștere a formelor se pot urma două căi:

	Caracter				Vârstă			Formă
	flegmatic	melancolic	sangvinic	Coleric	tânăr	matur	bătrân	X
Pers.1	1	0	0	0	0	1	0	(1,0,0,0,0,1,0)
Pers.2	0	1	0	0	1	0	0	(0,1,0,0,1,0,0)
Pers.3	1	0	0	0	1	0	0	(1,0,0,0,1,0,0)
Pers.4	0	0	1	0	0	0	1	(0,0,1,0,0,0,1)

Tabelul 4: Valorile caracteristicilor pentru un grup de patru persoane

Russel & Rao	$\frac{a}{p} = \frac{a}{a+b+c+d}$
Kendall	$\frac{a+d}{p} = \frac{a+d}{a+b+c+d}$
Yule	$\frac{ad-bc}{ad+bc}$
Ochiai	$\frac{a}{(a+c)\cdot(b+d)}$
Dice	$\frac{a}{a+2(b+c)}$

Tabelul 5: Exemple de indici de similaritate pentru forme cu caracteristici binare

- conceperea unei soluții viabile în funcție de particularitățile problemei, simplă și intuitivă, care să conducă la obținerea unor performanțe acceptabile;
- utilizarea formalismului matematic al unor metode specifice învățării supravegheate sau nesupravegheate, care pot asigura obținerea unui clasificator optim în funcție de anumite considerente.

De multe ori, o soluție simplă poate oferi obținerea rezultatelor dorite, fără a mai apela la metode specializate. În continuare va fi prezentată o astfel de abordare, considerând exemplul clasic al recunoașterii literelor ale alfabetului. Se va alege o reprezentare a formelor, se va construi un set de învățare și se va imagina un algoritm simplu privind clasificarea.

În conformitate cu discuțiile legate de figura 1, pentru reprezentarea fiecărui caracter se va utiliza un vector care are $9 \times 8 = 72$ de elemente și este construit pe baza împărțirii dreptunghiului în care este înscrisă imaginea care reprezintă caracterul în 9×8 dreptunghiuri de arii egale, după cum este exemplificat în figurile 3, 4 și 5. În funcție de numărul de pixeli aparținând caracterului aflați în interiorul fiecăruia din cele 9×8 dreptunghiuri, vectorii formă vor prezenta valorile binare 1 sau 0 cu semnificațiile de dreptunghi "ocupat" sau "neocupat".

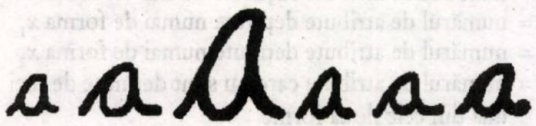


Figura 3: Diferite imagini reprezentând caracterul "a"

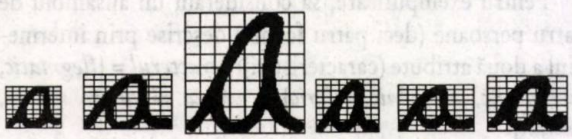


Figura 4: Normarea imaginilor din figura 3 folosind dreptunghiuri de dimensiune 9 x 8

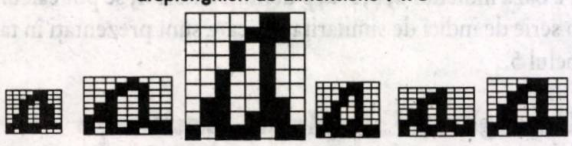


Figura 5: Reprezentarea finală a caracterelor folosind valori binare

Astfel, pentru al treilea caracter din șirul celor șase caractere prezentate în figura 5 vom avea următoarea reprezentare:

```
0 0 0 0 0 1 1 0
0 0 0 0 1 1 1 0
0 0 0 1 0 1 0 0
0 0 0 1 0 1 0 0
0 0 1 0 0 1 0 0
0 0 1 0 0 1 0 0
0 1 1 0 0 1 1 0
0 1 1 0 1 1 1 0
1 0 1 1 1 0 1 1
```

$a_3 = (0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1)$

O dată aleasă o reprezentare a formelor, următorul pas îl constituie construirea clasificatorului sau algoritmului de clasificare. Se va construi o mulțime de exemple (sau set de învățare) alcătuită din câteva reprezentări pentru fiecare caracter de recunoscut (de exemplu, pentru caracterul `"a"` se pot folosi cele șase reprezentări din figura 5).

Recunoașterea unui caracter nou, reprezentat de asemenea printr-un vector de 9×8 elemente, se poate face prin compararea acestuia cu fiecare formă din mulțimea de exemple având ca rezultat obținerea unui grad de asemănare. Cele mai simple moduri de definire a gradului de asemănare constau în:

- determinarea numărului "potrivirilor" dintre elementele 1 și 0 ale formei de recunoscut și ale formei comparate, urmărindu-se obținerea unui maxim;
- determinarea numărului "dezacordurilor" dintre elementele 1 și 0 ale formei de recunoscut și ale formei comparate, urmărindu-se obținerea unui minim.

În acest fel, se definește o distanță sau o măsură de similaritate dintre două forme. Evident, se poate folosi discuția privind similaritățile dintre formele cu caracteristici binare prezentată anterior.

Algoritmul de clasificare pentru o formă nouă este prezentat în continuare. Am considerat că mulțimea exemplelor conține M forme reprezentate sub forma unor vectori cu $N_1 \times N_2$ elemente (pentru cazul discutat anterior, avem $N_1 = 9$, $N_2 = 8$) desemnate prin variabilele f_i ($i = 0, ..., M - 1$).

Clasa formei f_i este dată de elementul de indice i al șirului `clasa` (care poate fi un șir de caractere). S-a utilizat prima modalitate de comparare prezentată și anume identificarea "potrivirilor". Forma necunoscută este reprezentată de vectorul `x`.

```
max ← 0
clasa_x ← 0
pentru i ← 0, M - 1 execută
  temp ← 0
  pentru j ← 0, N1 * N2 - 1 execută
    dacă x[j] = fi[j] atunci
      temp ← temp + 1
    sfârșit dacă
  sfârșit pentru
  dacă temp > max atunci
      max ← temp
      clasa_x ← i
 sfârșit dacă
sfârșit pentru
scrie clasa[clasa_x]
```

Acest exemplu simplu a fost discutat atât pentru a prezenta etapele unui demers într-o aplicație de recunoaștere a formelor, cât și pentru a accentua faptul că o metodă simplă și intuitivă și care permite obținerea unor rezultate cu o precizie acceptabilă poate fi utilizată uneori în locul unor metode mai avansate.

Radu-Daniel Vatavu este student în anul IV la Universitatea Ștefan cel Mare din Suceava. Poate fi contactat prin e-mail la **raduvro@yahoo.com**.

TEOREME de programare

Clara Ionescu

În prima fază a analizei unei probleme de algoritmică, de regulă, căutăm să stabilim clasa de probleme în care ar putea fi încadrată aceasta. Pentru un începător această etapă este dificilă, deoarece nu există încă o "arhivă" mentală în care poate avea loc căutarea. Vom expune, pe parcursul a câtorva articole teoremele de programare, încercând astfel să oferim un mod posibil de clasificare a problemelor, precum și a rezolvării lor eficiente.

Prelucrarea unui șir de date

O clasă importantă de probleme cu care ne întâlnim frecvent, cere ca pe baza unui șir de date de intrare să determinăm un singur rezultat. Vom începe tratarea primului subiect cu prezentarea unor probleme!

P1. Cunoscând mediile semestriale ale celor *n* elevi dintr-o clasă, să se determine media semestrială a clasei!

P2. Să se construiască o variabilă de tip *string* formată din *m* caractere date!

P3. Cercetând lumea păsărilor din Delta Dunării *k* specialiști au notat, fiecare, tipurile de păsări observate. Să se stabilească speciile de păsări din Delta Dunării!

P4. Să se calculeze produsul primelor *n* numere naturale (factorialul)!

Să analizăm problemele de mai sus și să încercăm să stabilim ce anume au în comun! În fiecare avem mai multe date de intrare de același tip, deci ele formează un șir (vector) și fiecare cere obținerea unui singur rezultat. Acest rezultat se poate determina prin aplicarea unei funcții care prelucrează întreg șirul de date (suma a *n* numere, concatenarea a *m* caractere, reuniunea a *k* mulțimi, produsul a *n* numere). Dar această funcție poate fi descompusă într-un șir de funcții, astfel încât argumentele lor să fie două entități (suma a două numere, concatenarea a două caractere, reuniunea a două mulțimi, produsul a două numere).

În proiectarea oricărui algoritm există o fază preliminară în care se stabilesc variabilele și semnificațiile lor:

n: *întreg* { *numărul elementelor șirului de prelucrat* }
x: *tablou(1..n): tip element* { *elementele șirului dat* }
F0: *tip element* { *element nul față de operație* }
S: *tip element* { *rezultatul* }

Aici, prin operație se înțelege oricare din operațiile realizabile cu instrumente matematice: adunare, produs, reuniune, intersecție, operații logice, concatenare etc. Oricare dintre aceste operații poate fi redusă la o operație binară și se poate stabili pentru ea un element neutru, astfel încât operația efectuată asupra unui element *y*, ales arbitrar și acesta are ca rezultat *y*.

Exemple

$F = \Sigma, \prod, \cup, \cap, \wedge, \vee$, & (concatenare)

F0 = 0, 1, {}, mulțimea valorilor posibile, Adevărat, Fals, '' (șirul vid).

Rezolvarea se construiește prin inducție matematică folosindu-se operația binară și elementul nul.

- dacă prelucrăm 0 elemente, rezultatul este *F0*,
- dacă se cunoaște rezultatul prelucrării primelor $i-1$ elemente (F^{i-1}), atunci rezultatul prelucrării primelor i elemente (F^i) se calculează astfel: $F^i = f(F^{i-1}, x_i)$.

Funcția *F*, exprimată recursiv este:

$$F_i := F(X_1, \ldots, X_i) := \begin{cases} F_0 & \text{, dacă } i = 0 \\ f(F^{i-1}, x_i), & \text{altfel} \end{cases}$$

Algoritmul în pseudocod este:

```
subalgoritm Prelucrare șir(n,x,S):
  S ← F0
  pentru i ← 1, n execută
    S ← f(S,x[i])
  sfârșit pentru
sfârșit subalgoritm
```

Vom aplica această rezolvare generală pentru problemele enunțate la începutul articolului, dând implementările în limbajul *Pascal*.

P1. Prima problemă cere determinarea unei medii aritmetice. Nu vom număra termenii sumei *S*, deoarece toate elementele șirului participă la sumă, în concluzie se cunoaște numărul cu care aceasta se împarte pentru a calcula media.

```
procedure P1(n:Byte;x:note;var medie:Real);
{ există definit tipul note = array[1..MaxN] of Real}
var i:Byte; S:Real;
begin
  s:=0;      { element neutru pentru adunare }
  for i:=1 to n do
    S:=S+x[i];      { operatia este adunarea }
  medie:=s/n
end;
```

P2. Cea de-a doua problemă cere concatenarea unor caractere într-o variabilă de tip *string*. Rezultă că operația va fi `'+'`, iar elementul neutru va fi șirul vid: `''`.

```
procedure P2(m:Byte;x:car;var S:string);
{ există definit tipul car = array[1..MaxM] of Char}
var i:Byte;
begin
  s:='';{ element neutru pentru concatenare }
  for i:=1 to m do
    S:=S+x[i];                    { concatenare }
end;
```

P3. În această problemă se cere determinarea reuniunii a *m* mulțimi. În *Pascal* există posibilitatea declarării unui tip utilizator *mulțime*, respectând restricția ca tipul de bază să fie ordinal. Pentru a nu încărca prezentarea cu elemente de programare inutile, vom considera că tipurile de păsări sunt notate cu numere naturale mai mici decât 255. Operatorul cu care implementăm reuniunea este `'+'`, iar elementul neutru este mulțimea vidă: `[]`.

```
procedure P3(k:Byte;x:mult;var S:o_multime);
  { există definite tipurile o_multime = set of Byte;}
    { și mult = array[1..Maxk] of set of Byte;}
var i:Byte;
begin
  s:=[];   { element neutru pentru reuniune }
  for i:=1 to k do
    S:=S+x[i];                    { reuniune }
end;
```

P4. Această problemă pare să fie cea mai simplă. Rezolvarea este similară cu prima, dar se finalizează după determinarea produsului.

```
procedure P4(n:Byte;var f:Integer);
var i:Byte;
begin
  f:=1;  { element neutru pentru inmultire }
  for i:=1 to m do
    f:=f*i;                       { inmultire }
end;
```

Ne ocupăm de această problemă, deoarece în multe manuale se pot vedea diferite enunțuri în care este nevoie de calcularea factorialului, dar rezolvările nu tratează detaliile de implementare "interesante" care se leagă de această problemă. În cele ce urmează vom intra în... laborator.

Evitarea depășirii

Să ne imaginăm un laborator într-o școală unde profesorul cere elevilor să scrie programul care calculează produsul primelor *n* numere naturale. Elevii scriu programul și fiind curioși, îl verifică. Introduc pe rând valori pentru *n*: 5, 8, ..., 25, ... poate și 50. În funcție de tipul variabilei declarate pentru rezultat se pot întâmpla lucruri intersante. De exemplu, dacă rezultatul este declarat ca fiind de tipul `Byte`, elevul respectiv obține că `6!=208`, iar `8!=128`. Dar orice elev știe că `8!` trebuie să fie mai mare decât `6!`. Încep căutările disperate în programul sursă. Unde am greșit? se întreabă elevul nostru. Să presupunem în continuare că își dă seama că trebuie să schimbe tipul rezultatului. Va efectua modificarea tipului rezultatului din `Byte` în `Word` sau direct în `Longint`. În acest caz, elevul nostru devine curajos, rulează programul pentru $n = 20$, apoi pentru $n = 50$. Și din nou surpriză: în cazul declarării rezultatului de tip `Longint`, `20!=-2102132736` (număr negativ!), iar `50!=0`. Până la urmă și cel mai răbdător începător se enervează și își declară variabila respectivă de tipul `Real`. În schimb, acum programul lui se întrerupe pentru $n > 33$ cu mesajul de eroare: `Floating point overflow`! Ce-i de făcut?

Să luăm în considerare situația în care se preferă ca rezultatul să fie de tip `Longint`. Evident, încercările ne-au convins, că începând cu o anumită valoare a lui *n* se produce depășire. Mai ales atunci când trebuie calculat factorialul unui *n*, care constituie un rezultat calculat în program, ne trebuie "un instrument" care să ne protejeze de neplăceri, cum ar fi obținerea unui rezultat fals. Cel mai mare număr de tipul `Longint` posibil de reprezentat în calculator este `2147483647`, reținut de constanta simbolică a *unit*-ului *System* numită `MaxLongint`. Ar urma ca atare să putem pune o întrebare de genul `f > MaxLongint`... Dar este absurd să întrebăm dacă un număr este mai mare decât cel mai mare număr posibil...

În consecință, vom deveni preventivi și vom pune întrebarea cu un pas înainte de efectuarea înmulțirii curente care oricând ar putea produce o depășire în felul următor:

```
procedure P4_bis(n:Byte;var f:Longint);
var i:Byte;
begin
  f:=1; i:=1;
  while (f<>0) and (i<=n) do begin
    if f>MaxLongint div i then f:=0
    else begin f:=f*i; Inc(i) end
  end
end;
```

Valoarea lui `f` se va testa în blocul apelant pentru a decide dacă s-a putut calcula factorialul cerut, sau determinarea lui s-a întrerupt deoarece s-ar fi produs o depășire.

Noul HP

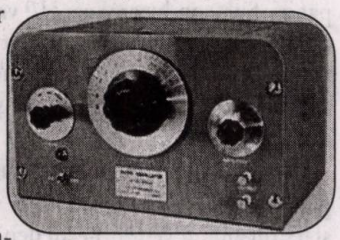
Carmen Șușman

Fuziunea dintre HP și COMPAQ a fost unul dintre cele mai importante evenimente care au avut loc în industria IT&C. În cadrul acestui articol vom prezenta istoria celor două companii care au participat la fuziune.

Hewlett-Packard Company

După ce au absolvit cursurile *Universității Standford* în 1934, ***Bill Hewlett*** și ***Dave Packard*** au mers împreună într-o excursie, timp în care au devenit buni prieteni. După aceea, *Bill* și-a continuat studiile la *MIT* și *Standford*, iar *Dave* s-a angajat la *General Electric*. După puțin timp, încurajați de ***Fred Terman***, profesor la *Standford* și mentorul celor doi, ei au decis să își creeze propria afacere și să se ocupe singuri de ea.

În anul 1938, *Dave* și soția sa s-au mutat în localitatea *Palo Alto* din *California*. În spatele locuinței lor era o altă casă care a fost închiriată de *Bill*. *Dave* și *Bill* au început să lucreze într-un garaj, având la dispoziție un capital de 538$.



HP 200A, primul produs

Pe baza cunoștințelor lui *Bill*, a fost construit primul produs al companiei *HP*, oscilatorul audio *HP 200A*, un instrument electronic folosit pentru testarea echipamentelor de sunet. *Dave* spunea că denumirea produsului a fost aleasă pentru a da impresia că *HP* este o companie mai veche care a oferit și alte produse. *Walt Disney* a comandat opt oscilatoare *HP 200B* pentru a le folosi la dezvoltarea coloanei sonore pentru *Fantasia*.

Compania a apărut oficial pe data de 1 ianuarie 1939. Pentru a decide numele acesteia (*Hewlett-Packard* sau *Packard-Hewlett*) s-a folosit cea mai bună metodă posibilă: ***cap sau pajură***; se pare că *Bill* a avut mai mult noroc. În acest an compania a avut un venit de 5369$.

Anii '40

Calitatea produselor *HP* a început să fie recunoscută de ingineri și oameni de știință. Începutul celui de-al doilea război mondial a determinat guvernul *SUA* să comande o mulțime de dispozitive electronice.

HP a profitat de situație și a reușit să își construiască prima clădire și să lanseze noi produse. În 1940 sediul companiei s-a mutat din garaj în spații închiriate din *Palo Alto*.

Al doilea sediu HP

La puțin timp după încheierea primului an de funcționare, *Bill* și *Dave* au hotărât că toți angajații ar trebui să poată beneficia direct de profiturile companiei. Astfel, angajații au primit de *Crăciun* un bonus de 5$. De asemenea, în iunie a avut loc prima donație caritabilă a companiei; valoarea acesteia a fost de 5$.

În 1941 *Bill* a devenit ofițer în armata *SUA* și *Dave* a condus singur compania până în 1947.

În 1942 a început construcția primei clădiri aflate în proprietatea companiei. Aceasta a fost concepută în așa fel încât să poată fi foarte ușor transformată într-o băcănie, dacă actuala afacere nu s-ar mai fi dezvoltat.

În 1943, *HP* a început să producă generatoare de semnal (care foloseau microunde) pentru *Laboratorul de Cercetare Naval*. După război, s-a luat decizia să se continue lucrul în domeniul microundelor. Acest domeniu a ajuns, în scurt timp, una dintre cele mai importante preocupări ale companiei și *HP* a devenit lider pe piața generatoarelor de semnal.

Între 1939 și 1943 veniturile companiei au crescut de la 5369$ la 953294$, iar numărul angajaților de la 2 la 45.

Pe data de 18 august 1947, *HP* devine corporație; *Dave* este numit președinte, iar *Bill* vicepreședinte. *HP* oferea 39 de produse și *Dave* spunea că viitorul companiei este foarte promițător.

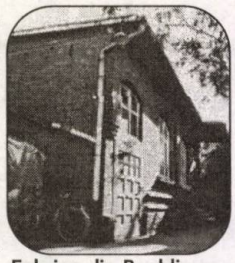
Până în 1949 compania a ajuns să aibă 166 de angajați, iar veniturile anuale depășeau 2200000$.

Anii '50

În 1951 a fost lansat produsul *HP 524A*, un numărător de frecvență care funcționa la o viteză mult mai mare decât alte produse asemănătoare. Timpul necesar măsurării frecvențelor înalte a fost redus de la 10 minute la 2 secunde. De-a lungul anilor, produsele de acest tip au adus companiei venituri de miliarde de dolari. În 1956 compania a produs primele sale osciloscoape.

În 1957 compania a ajuns să fie cotată la bursă, valoarea unei acțiuni fiind de 16$. Un an mai târziu are loc pri-

ma achiziție; *HP* a cumpărat *F.L. Moseley Company* din *Pasadena* și astfel a ajuns pe piața *plotter*-elor. În acest mod s-au pus bazele viitoarelor afaceri cu imprimante.


Fabrica din Boeblingen

În 1959 *HP* a devenit o companie internațională prin deschiderea unei filiale în *Geneva* (*Elveția*) și a unei fabrici în *Boeblingen* (*RFG*). La sfârșitul anului *HP* a avut venituri de 48 de milioane de dolari și 2378 de angajați.

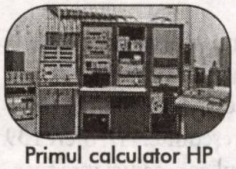
Anii '60

În 1961 *HP* achiziționează compania *Sanborn*, specializată în produse medicale. În anul 2000 linia de produse medicale va fi preluată de *Agilent Technologies*, companie desprinsă din *HP*. Un an mai târziu compania *HP* ajunge una dintre primele 500 din *Statele Unite*, potrivit revistei *Forbes*.

În 1963 se intră pe piața asiatică prin intermediul primului parteneriat. La *Tokyo* apare *Yokogawa-Hewlett-Packard* ca urmare a colaborării dintre *HP* și *Yokogawa Electric Works*. Apare sintetizatorul de frecvențe 5100*A*, unul dintre cele mai complexe produse oferite de *HP* în întreaga sa istorie. Acesta era capabil să reproducă cu o mare acuratețe sunetele unei game foarte largi de instrumente muzicale.

În 1964 compania își celebrează cea de-a 25-a aniversare; *Dave Packard* devine *CEO* și *chairman* al *Consiliului de Administrație*, iar *Bill Hewlett* este ales președinte al companiei. Apare ceasul atomic *HP* 5060*A* care era capabil să măsoare timpul cu erori de numai o secundă la 3000 de ani.

Achiziția companiei *F&M Scientific Corporation* din 1965 permite îmbunătățirea produselor de măsurare și testare cu aplicații în domeniul analizei chimice.


Primul calculator HP

În 1966 apare primul calculator *HP*; produsul a fost cunoscut sub denumirea *HP* 2116*A* și a fost primul pentru care *HP* a folosit circuite integrate. Având între 4KB și 32 KB de memorie bazată pe medii magnetice, calculatorul costa între 25000$ și 50000$, în funcție de opțiuni.

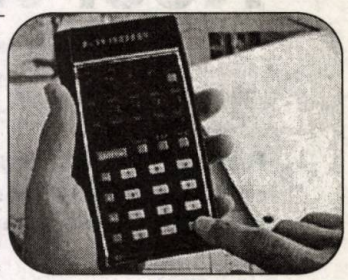
Tot în acest an, *Lyndon B. Johnson*, președintele *SUA* din acel moment, îl numește pe *Bill Hewlett* membru în *Comitetul Național de Consultare Științifică*.

Primul calculator programabil din lume a fost *HP* 9100*A*, lansat în 1968.

În 1969 *Dave Packard* devine adjunctul *Secretarului de Stat al Apărării*, funcție pe care a deținut-o până în 1971. În același an *HP* lansează primul său sistem de operare bazat pe divizarea timpului, care permitea accesul simultan a 16 utilizatori. La sfârșitul anului profitul companiei a fost de 326 de milioane de dolari, iar numărul angajaților a ajuns la 15840.

Anii '70

În 1971 *HP* a produs un instrument de măsură capabil să efectueze măsurători cu precizii de ordinul fracțiunilor de microni. Un an mai târziu, *Dave* renunță la funcția de secretar adjunct al apărării și devine *chairman* al *Consiliului de Administrație* al companiei. *Bill* își păstrează titlurile de *CEO* și președinte al companiei. În acest an apare calculatorul științific de buzunar *HP*-35; în anul 2000 acest produs este numit unul dintre cele mai importante douăzeci de produse ale tuturor timpurilor. De asemenea, apare *HP* 3000, primul calculator produs de *HP*, destinat mai multor tipuri de utilizatori.


HP-35

În 1973 *HP* produce primul calculator capabil să afișeze caractere japoneze. Noul produs, *HP* 9810, este lansat pe piață în *Japonia* de către *Yokogawa-Hewlett-Packard*.

În 1974 *HP* lansează primul calculator care folosește memorie *DRAM* în locul celei bazate pe medii magnetice. Apare *HP*-65, primul calculator programabil de buzunar.

În anul 1976, *HP* obține pentru prima dată venituri de peste un miliard de dolari.

În 1977 *Bill* se retrage din funcția de președinte, dar o păstrează pe cea de *Chief Executive Officer*. *John Young*, unul dintre primii angajați *HP*, devine președinte al companiei. Sub conducerea sa, *HP* va ajunge lider în domeniul calculatoarelor. În același an, este lansat produsul *HP*-01, o combinație între un ceas, calendar personal și calculator. Un an mai târziu *Bill* se retrage și din funcția de *CEO*, titlul fiind preluat de *Young*.

În 1979 *Dave* pune bazele unei filiale chineze a companiei. *Hewlett-Packard China* va fi înființată în 1985. La sfârșitul anului veniturile au depășit 2,4 miliarde de dolari, iar numărul angajaților a fost de 52030.

Anii '80

Primul calculator personal produs de *HP* a fost *HP*-85, lansat în 1980. În același an a apărut și prima imprimantă *LASER* oferită de această companie. Doi ani mai târziu compania a lansat *HP*-75, primul său calculator *handheld*.

HP-85

În 1983 a fost lansat *HP*-150, un calculator personal care folosea *touchscreen*-uri. În acest an *Bill Hewlett* a primit *Medalia Națională pentru Știință*, cea mai importantă distincție națională pe care o putea primi un om de știință american.

Anul 1984 a fost cel în care au apărut primele imprimante *HP* cu jet de cerneală. Produsul a fost denumit *HP ThinkJet* și oferea o rezoluție de 96 *dpi*, costuri scăzute și calitate mult superioară față de cea a imprimantelor cu ace.

Tot în acest an a fost lansată familia *HP LaserJet*; aceste imprimante ofereau o rezoluție de 300 *dpi*.

În 1987 *Bill Hewlett* se retrage din funcțiile de conducere pe care le deținea și primește titlul onorific de *director emeritus*.

În 1988 apar imprimantele *HP DeskJet*, prima familie de imprimante cu jet de cerneală destinate publicului larg. *HP* devine una dintre cele mai importante 50 de companii ale lumii.

În 1989 a fost sărbătorită aniversarea de 50 de ani de la nașterea companiei. Veniturile au ajuns la 11,9 miliarde de dolari pe an, iar compania avea 95000 de angajați.

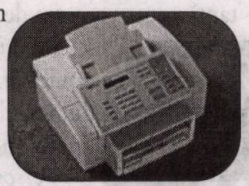
Anii '90

În 1991 *HP* introduce prima sa imprimantă color cu jet de cerneală (*HP DeskJet* 500C), ceea ce duce la o adevărată revoluție în domeniul listărilor color.

Anul 1992 este cel în care *John Young* își anunță retragerea. Conducerea companiei este preluată de ***Lew Pratt***, angajat *HP* începând cu anul 1966.

În 1993 *HP* introduce un calculator portabil care cântărea doar 1,5 kg. În același an este vândută imprimanta *HP LaserJet* cu numărul 10.000.000, iar *Dave Packard* se retrage și el din funcțiile de conducere.

În 1994 începe colaborarea cu *Intel* pentru crearea unei arhirecturi de 64 de biți pentru procesoarele secolului XXI. Efortul se concretizează în anul 2001 când a apărut procesorul *Intel Itanium*. Este lansat produsul *HP OfficeJet*, o combinație între o imprimantă, un fax și un copiator.



HP OfficeJet

În 1995 *Dave Packard* publică ***The HP Way***, o carte care prezintă nașterea companiei și explică modul în care a ajuns o companie de succes. La un an după publicare, pe data de 26 martie 1996, are loc decesul lui *Dave Packard*.

În 1997 *HP* obține premiul *Emmy* pentru contribuțiile aduse la dezvoltarea tehnologiei de compresie *MPEG*.

În 1999 se formează *Agilent Technologies*; *HP* deține 84,1% din acțiunile noii companii. În iulie *Lew Pratt* se retrage, iar în funcția de președinte și *CEO* al companiei este numită ***Carleton (Carly) S. Fiorina***. Sub conducerea ei, *HP* dezvoltă o strategie pentru era *Internet*-ului, bazată pe servicii *Web*. La sfârșitul anului, *HP* a anunțat venituri de 42 de miliarde de dolari și un număr de 84400 de angajați.

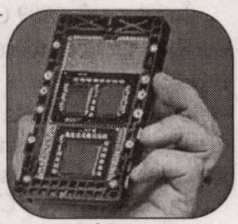
Ultima perioadă

În anul 2000 *HP* a ajuns pe poziția a 13-a în topul celor mai importante companii. În luna iulie, *HP* renunță la acțiunile *Agilent Technologies* care devine o companie complet independentă.

Pe data de 22 septembrie, *Carly Fiorina* primește și funcția de *chairman* al *Consiliului de Administrație* al companiei *HP*.

Pe data de 12 ianuarie 2001, se stinge din viață și *Bill Hewlett*, unul dintre cei doi fondatori ai companiei.

În luna mai apar primele produse *HP* care folosesc procesorul *Intel Itanium*.



Intel Itanium

Pe data de 4 septembrie este anunțat faptul că s-a ajuns la un acord definitiv de fuziune între *HP* și *COMPAQ*.

Fuziunea se finalizează pe data de 3 mai 2002; noua companie valorează 87 de miliarde de dolari, are peste un miliard de clienți în 162 de țări și un număr de peste 150000 de angajați. Noua companie păstrează denumirea *Hewlett-Packard*.

Compaq Computer Corporation

Compania ***Compaq*** a fost fondată în februarie 1982 de ***Rod Canion***, ***Jim Harris*** și ***Bill Murto***, trei angajați de la *Texas Instruments*. Cei trei au investit fiecare câte 1000$ pentru a crea o nouă companie; primul produs a fost un calculator personal portabil, schița sa fiind creată într-o plăcintărie din *Houston*.

Fondatorii și-au prezentat ideile în fața lui ***Ben Rosen***, președinte la *Sevin-Rosen Partners*, o companie de plasare a capitalului. Acesta a fost impresionat de ideea unui calculator portabil și a decis să finanțeze noua companie devenind *chairman* al *Consiliului de Administrație*. Și-a păstrat funcția până în septembrie 2000 când ***Michael Cappelas*** a fost ales ca *chairman* prin vot unanim.

Primul președinte și *CEO* al companiei *Compaq* a fost *Rod Canion*, unul dintre cei trei fondatori. Pe data de 4 noiembrie a fost lansat primul produs, calculatorul portabil *Compaq Portable PC*, primul astfel de produs disponibil pe piață. Acesta folosea un procesor cu frecvența de 4,77 MHz, avea două unități *floppy* de 5,25 inci, o memorie *RAM* de 640 KB și cântărea aproape 13 kg.

În primul an de activitate *Compaq* a vândut peste 53000 de calculatoare portabile și a realizat un venit de 111,2 milioane de dolari, un record pentru primul an în întreaga istorie a *Statelor Unite*. Prima ofertă publică de acțiuni *Compaq* pe piața *Nasdaq* a dus la obținerea a 67 de milioane de dolari.

În 1984 apar primele *desktop*-uri produse de *Compaq*. *PC*-urile *Compaq* sunt lansate și în Europa. Al doilea an de activitate aduce venituri de 329 de milioane de dolari, un alt record în istoria *Statelor Unite*. Un alt record este obținut pentru al treilea an de activitate, valoarea vânzărilor fiind de 503,9 milioane de dolari.

În aprilie 1986 *Compaq* devine una dintre cele mai importante 500 de companii la doar patru ani după înființare, stabilind un nou record. În această lună este vândut *PC*-ul *Compaq* cu numărul 500000. *PC*-ul cu numărul 1000000 este vândut în noiembrie 1987.

Seria recordurilor continuă în februarie 1988 când sunt anunțate venituri de 1,2 miliarde de dolari pentru anul 1987. *Compaq* a fost prima companie care a reușit să vândă pro-

duse în valoare de peste un miliard de dolari la doar șase ani după lansare.

În februarie 1989 *Compaq* devine cel de-al doilea producător de calculatoare din *Europa*, depășind mult mai cunoscutele firme *Apple* și *Olivetti*. În octombrie este lansat *Compaq LTE*, primul *notebook* oferit de această companie. De asemenea, în noiembrie apare și *Compaq Systempro*, primul *server* lansat pe piață de *Compaq*.

În ianuarie 1991 *Compaq* anunță că a reușit să obțină pentru prima dată venituri de un miliard de dolari într-un singur trimestru. În septembrie este lansată familia de produse *Compaq Deskpro/M*.

În iunie 1992 *Compaq* intră pe piața calculatoarelor ieftine oferind produsele *ProLinea* și *Contura*. În octombrie apar și primele servere cu prețuri reduse, familia de produse fiind *Compaq ProSignia*.

În mai 1993 *Compaq* începe să folosească procesoarele *Intel Pentium* pentru calculatoarele sale. În iulie departamentul de *PC*-uri este divizat în două departamente distincte, unul pentru *desktop*-uri și unul pentru *notebook*-uri. Luna septembrie marchează debutul seriei de produse *Compaq Presario*. Două luni mai târziu apar primele calculatoare *minitower*.

În ianuarie 1994 *Compaq* decide să ofere calculatoare care folosesc procesoare produse de *AMD*. În luna aprilie analiștii afirmă că, în primul trimestru al anului, numărul unu mondial pe piața calculatoarelor a fost *Compaq*.

În ianuarie 1995 *Compaq* ajunge să dețină cel mai mare segment din piața *PC*-urilor. Un an mai târziu sunt anunțate venituri de 14,8 miliarde de dolari; *Compaq* ajunge pe poziția a 5-a între companiile de calculatoare din întreaga lume.

Cei trei fondatori ai companiei Compaq

În martie 1996 *Compaq* anunță *LS*-120, prima unitate *floppy* cu capacitatea de 120 MB. În iunie este lansată familia de *notebook*-uri ieftine *Armada*. În noiembrie este vândut *server*-ul *Compaq* cu numărul 1000000.

În ianuarie 1997 este anunțat *Compaq Presario* 1060*ES*, primul calculator portabil proiectat pentru a fi folosit în scopuri educaționale. O lună mai târziu apar primele calculatoare *Compaq* al căror preț este mai mic decât 1000$ (seria *Presario* 2000).

În ianuarie 1998 sunt anunțate venituri anuale de 24,6 miliarde de dolari. *Compaq* primește titlul de companie a anului 1997, acordat de revista *Forbes*. În luna mai *Compaq* anunță că oferă imprimante color cu jet de cerneală.

În ianuarie 1999 *Compaq* creează compania *AltaVista*. În aceeași lună este lansată cea mai rapidă stație de lucru din lume: *AlphaPowered™ Professional Workstation XP*10. În iulie 1999 *Michael D. Capellas* devine președinte și *CEO* al companiei.

În martie 2000 *Compaq* lansează calculatoarele *Presario Internet* care folosesc procesoare cu frecvența de 1 GHz. Un alt produs original al companiei, *iPAQ Pocket PC*, este lansat în luna aprilie.

În iunie 2001 este vândut calculatorul *iPAQ Pocket PC* cu numărul 1000000. În aceeași lună *Compaq* începe să folosească procesoarele *Intel Itanium* pentru serverele pe 64 de biți. În luna iulie *Compaq* anunță începerea fuziunii cu *Hewlett-Packard*.

Pe data de 3 mai 2002 compania *Compaq* își încetează existența, devenind parte a companiei *HP*, procedurile legale necesare fuziunii fiind finalizate.

Noul HP

La patru zile după finalizarea fuziunii din data de 3 mai 2002, a avut loc lansarea oficială a noii companii *Hewlett-Packard*. Așadar, noul *HP* a început să își desfășoare activitatea pe data de 7 mai 2002. Foștii acționari *HP* dețineau acum 64% dintre acțiunile noii companii, în timp ce foștii acționari *Compaq* erau proprietarii restului de 36%. *Carly Fiorina* a continuat să conducă *HP*, având funcțiile de *CEO* și *chairman*. *Michael Capellas*, fostul *CEO* al companiei *Compaq*, a devenit președinte al noului *HP*.

Compania *HP* este organizată în patru grupuri de afaceri, fiecare dintre ele ocupându-se de produse care se adresează unui anumit tip de clienți.

Enterprise Systems Group se ocupă de *server*-e, stocarea datelor, tehnologii de *networking* și oferirea de soluții de infrastructură.

HP Services este grupul care oferă servicii de consultanță pentru crearea de infrastructuri *IT*.

Imaging and Printing Group are ca domeniu de activitate oferirea de soluții în domeniul imprimantelor, al imaginii digitale, sistemelor digitale etc.

Personal Systems Group se ocupă de calculatoare personale, *notebook*-uri, dispozitive *handheld*, dispozitive pentru accesul la *Internet* etc.

În ceea ce privește accesul clienților la produsele preferate (oferite de fostul *HP* sau de *Compaq*) nu vor exista modificări. Aceștia vor putea să achiziționeze aceleași produse, de la aceleași megazine. Singura schimbare este faptul că foștii clienți *Compaq* au devenit acum clienți *HP*.

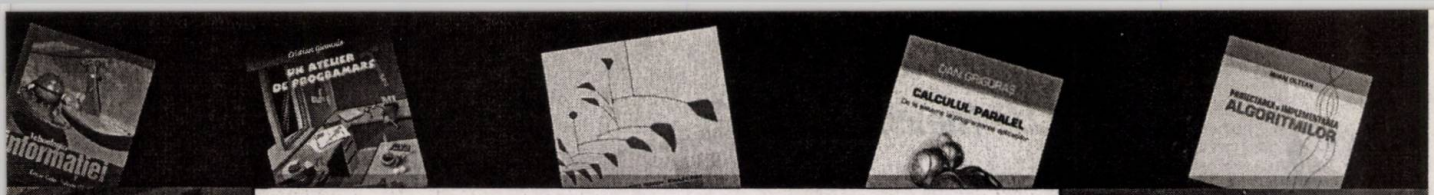
Unul dintre punctele forte ale fuziunii este faptul că produsele oferite de *HP* și *Compaq* erau, în mare măsură, complementare; astfel nu este nevoie ca producția unora să fie oprită.

Bibliografie

- **http://www.hp.com;**
- **http://www.compaq.com;**
- **http://thenew.hp.com.**

Carmen Șușman este redactor GInfo și poate fi contactată prin e-mail la adresa **schummy@email.ro.**

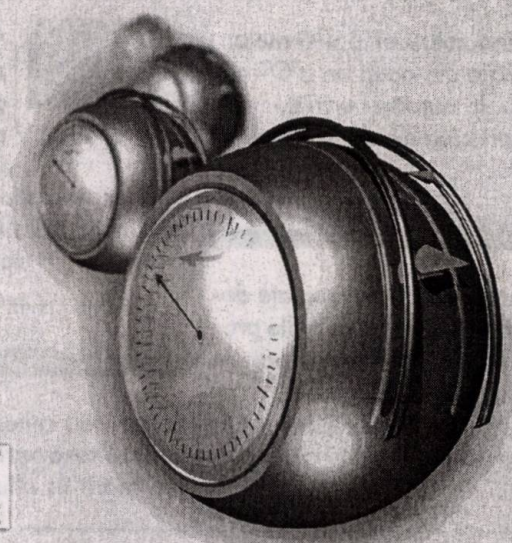
COMPAQ hp invent



DAN GRIGORAȘ

CALCULUL PARALEL

De la sisteme la programarea aplicațiilor



Calculul paralel. De la sisteme la aplicații tratează, într-un mod natural și structurat, cele mai importante aspecte ale proiectării aplicațiilor paralele, sau, cu un termen mai nou, de mare performanță. Paralelismul, care înseamnă executarea mai multor operații simultan, este o caracteristică a tuturor calculatoarelor actuale, inclusiv a celor personale. De aceea, cunoașterea principiilor și legilor care îl guvernează, ca și a metodologiilor pentru proiectarea aplicațiilor cele mai performante este o necesitate,dacă se dorește exploatarea optimă a resurselor sistemelor de calcul. Parametrii de performanță și metodologiile de proiectare și execuție optimă reprezintă subiectele care guvernează discursul științific al cărții de față.

COMPUTER LIBRIS AGORA

COMPUTER LIBRIS AGORA

Informații la tel. 064-192422

În numărul URMĂTOR

Numărul următor al revistei noastre va apărea în cursul lunii noiembrie. Câteva dintre articolele pe care le veți putea citi sunt prezentate pe scurt în continuare...

ONI 2002

Vom continua prezentarea soluțiilor problemelor propuse spre rezolvare la ediția din acest an a Olimpiadei Naționale de Informatică. În numărul următor veți găsi soluțiile câtorva dintre problemele de la clasele a XI-a și a XII-a.

Eureka

Acest articol se dorește a fi o prezentare succintă a unui rezolvitor de probleme. Va fi descris modul în care poate fi utilizat limbajul Eureka pentru rezolvarea anumitor probleme care ar necesita eforturi destul de mari pentru un utilizator care nu deține un bagaj impresionant de cunoștințe în domeniul matematicii.

Bursele Agora

În numărul următor al revistei vor fi prezentate detalii referitoare la modalitățile de rezolvare ale problemelor de la etapa finală a ediției anterioare a concursului de programare "Bursele Agora".

De asemenea, veți găsi informații referitoare la primele etape ale celei de-a patra ediții a concursului nostru.

Probleme propuse

La cererea cititorilor, ponderea problemelor propuse spre rezolvare în cadrul paginilor revistei noastre s-a mărit în ultima perioadă. În numărul din luna mai au fost prezente 40 de enunțuri, iar în acest număr 23. Vom păstra și în numărul din noiembrie acest "obicei". Sperăm că problemele vi se vor părea interesante.

Simularea paralelismului

De multe ori suntem puși în situația de a realiza un program cât mai simplu care să "pară" că folosește metode sofisticate cum ar fi paralelismul. În acest articol autorul își propune să prezinte modul în care pot fi simulați anumiți algoritmi paraleli simpli de sortare.

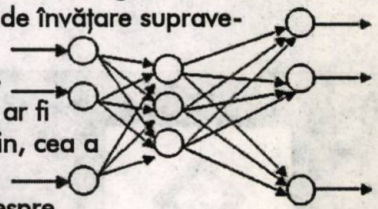
CEOI 2002

Enunțurile problemelor de la cea de-a noua ediție a Olimpiadei de Informatică a Europei Centrale au fost prezentate în acest număr la pagina 11.

Cititorii interesați de soluțiile oficiale ale acestor probleme vor putea găsi în numărul următor detaliile dorite.

Recunoașterea formelor

Serialul dedicat recunoașterii formelor grafice va continua cu prezentarea unor modalități de învățare supravegheată care sunt utilizate în acest domeniu. Vor fi descrise unele reguli de bază cum ar fi cea a celui mai apropiat vecin, cea a celor mai apropiați k vecini, funcții discriminant, detalii despre rețelele neuronale, regula lui Bayes etc.



GHIDUL TĂU ÎN LUMEA TEHNOLOGIILOR

PC MAGAZINE ROMÂNIA

de reținut!

O nouă modalitate de a face afaceri, prin internet!

Cursuri pentru studenți

4 - 15 noiembrie 2002

www.pcmagazine.ro, www.fwd.aiesec.ro

Misiunea noastră este de a testa și evalua produse și servicii din domeniul tehnologiei informației și comunicațiilor și de a prezenta rezultatele cu onestitate și obiectivitate

AGORA MEDIA Str. C-tin Rădulescu Motru nr. 13, etaj 4, apt. 42, sect. 4 (CP 94/49), București
Tel:021-3309282, Fax:021-3309285

Agora MEDIA

Publicarea acestei reviste este posibilă datorită sponsorizărilor acordate de următoarele firme care s-au decis să sprijine învățământul românesc de informatică

SPONSORI DE PLATINĂ

COMPAQ

SPONSORI DE AUR

FORTE COMPUTERS
Flexibility. Made in Romania

ASTRAL TELECOM

SPONSORI DE ARGINT

sat
IT SOLUTIONS & SERVICES

hp
invent

ERICSSON

Magna

M.G.T.
EDUCATIONAL

a&C
INTERNATIONAL S.A.
the CAD CAMpany
autodesk
authorized distributor

ALCATEL

*Rugăm companiile care doresc să se alăture acestui efort să ne contacteze prin e-mail la adresa office@ginfo.ro sau la numărul de telefon: 0265-166.516

Noiembrie 2002
Vol. 12/7

Lei 30.000

gazeta de informatică

Revista viitorilor specialiști în tehnologia informației

din sumarul numărului:

CEOI 2002

SOLUȚIILE problemelor

focus

Simularea paralelismului

babel

EUREKA

rezolvitor de probleme

Plus

Recunoașterea formelor-elemente de grafică-episodul 2
Probleme preluate din Gazeta de Informatică nr.4/1994
Olimpiada Națională de Informatică • Bursele Agora
Admiterea UBB 2002 • Teoreme de programare

GHIDUL TĂU ÎN LUMEA TEHNOLOGIILOR

PC MAGAZINE ROMÂNIA

de reținut!

CONCURS DE FOTOGRAFIE DIGITALĂ

HP România –
PC Magazine România

1 martie - 24 mai 2002

www.hp.com.ro, www.pcmagazine.ro

Misiunea noastră este de a testa și evalua produse și servicii din domeniul tehnologiei informației și comunicațiilor și de a prezenta rezultatele cu onestitate și obiectivitate

AGORA MEDIA Str. C-tin Rădulescu Motru nr. 13, etaj 4, apt. 42, sect. 4 (CP 94/49), București
Tel: 01-3309282, Fax: 01-3309285

www.pcmagazine.ro

Agora MEDIA

Vol. 12, Nr. 7 /2002

gazeta de informatică

Redacția:

Director științific:
prof. dr. *Horia Georgescu*
Director de publicație:
lect. univ. *Clara Ionescu*
Redactor șef:
Mihai Scorțaru
Redactor șef adjunct:
Carmen Șușman
Redactori:
prof. *Eugen Ionescu*
Claudiu Soroiu
Au mai colaborat:
Emilian-Marius Bold (Oțelu Roșu)
conf. dr. *Marin Vlada (București)*
prof. univ. *Militon Frențiu (Cluj)*
Radu-Daniel Vatavu (Suceava)
Tehnoredactare:
Claudiu Soroiu
Coperta și grafica:
Andrei Pașa

Adresa pentru corespondență:
Computer Libris Agora,
Universității 8/8,
3400, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 0264/192.422
0265/166.516
Fax: 0265/166.290
http://www.ginfo.ro
ISSN: 1222-7129

Tiparul: *Master Druck* Târgu Mureș
tel: 0265/162.359
Editor: Agora Media SRL
Agora Media SRL, editor al publicațiilor "*eWeek România*", "*PC Magazine România*", "*NetReport*" și "*Gazeta de Informatică*", este membru al Biroului Român pentru Auditarea Tirajelor.
str. T. Vladimirescu nr. 63/9
4300 Târgu-Mureș
Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director tehnic:
Szabó László
Vânzare reclamă:
Mihai Bucuroiu
Cosmin Ursu
Denis Marioara
Marius Comărniceanu
Reprezentanța București:
Director: *Dana Pastramă*
Tel: 021/330.92.81
Fax: 021/330.92.85

Sumar

probleme.. PAG. 5

AMINTIRI

~ probleme preluate din ~
~ Gazeta de Informatică ~
~ 4 / 1994 ~

soluții.. PAG. 8

Olimpiada NAȚIONALĂ de Informatică

~ soluțiile problemelor de la ~
~ clasele a XI-a și a XII-a ~

CEOI 2002

~ soluțiile problemelor ~
~ de la ediția 2002 a ~
~ Olimpiadei de Informatică ~
~ a Europei Centrale ~

concurs.. PAG. 22

Etapa FINALĂ

~ soluțiile oficiale ale ~
~problemelor de la etapa~
~ finală a ediției 2001/2002 ~

Bursele AGORA

~ enunțurile problemelor ~
~ propuse spre rezolvare ~
~ la primele patru etape~
~ ale ediției 2002/2003 ~

focus.. PAG. 30

Simularea PARALELISMULUI

Emilian-Marius BOLD

~ algoritmi paraleli de sortare ~

babel.. PAG. 33

ADMITERE UBB 2002

Militon FRENȚIU

EUREKA - rezolvitor de probleme

Marin VLADA

begin.. PAG. 39

TEOREME de programare

Clara IONESCU

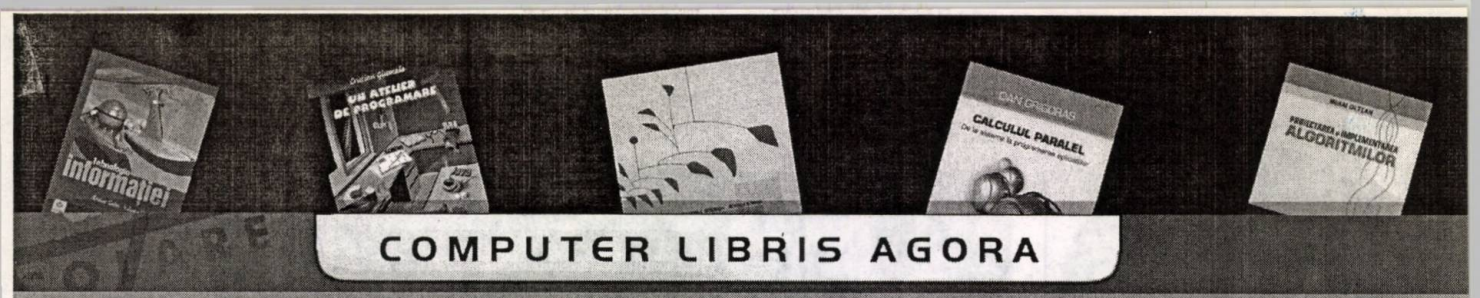
~ teorie și practică ~

serial.. PAG. 42

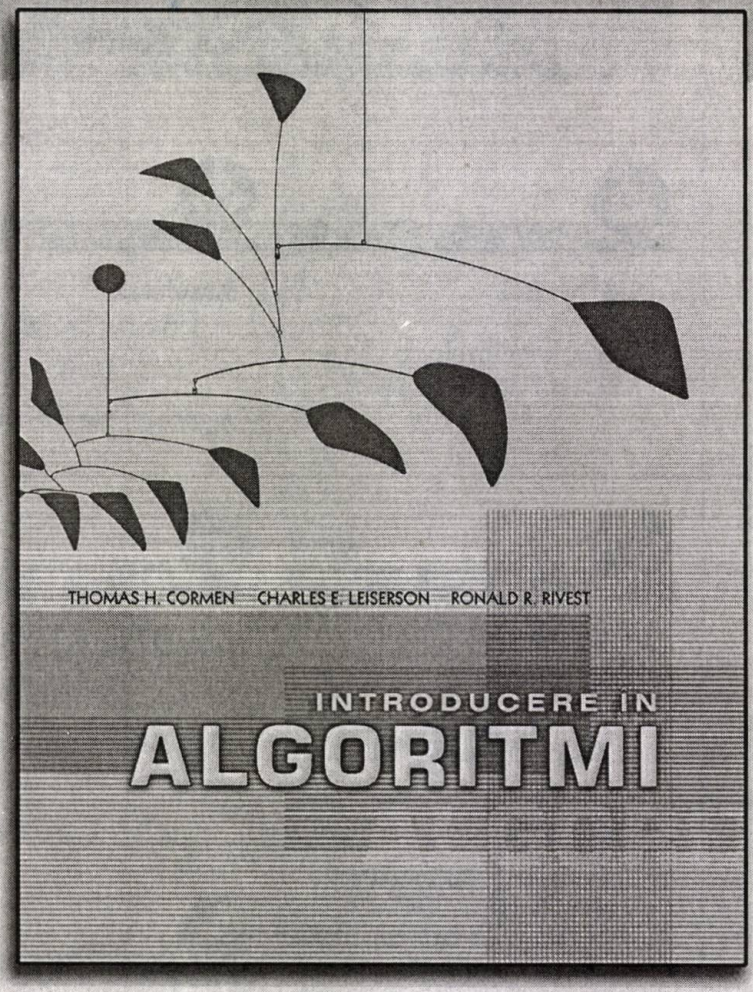
RECUNOAȘTEREA formelor

Radu-Daniel VATAVU

~ elemente de grafică ~
~ al doilea episod ~



COMPUTER LIBRIS AGORA



Cartea se adresează, în primul rând, celor care studiază la nivel elementar sau academic algoritmi și structuri de date. Deoarece tratează aspecte teoretice și practice, legate de proiectarea algoritmilor, este la fel de binevenită elevilor, studenților, profesorilor și autodidacților.
Introducere în algoritmi acoperă atât metodele clasice cât și metodele moderne ca
analiza amortizată și algoritmii paralleli.
Prezentarea matematică, riguroasă, este atent detaliată în așa fel încât să fie accesibilă tuturor cititorilor.
Capitolele încep cu elemente de bază, avansând spre subiecte mai complexe.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

AMBIȚII

Oare există elevi și studenți care nu s-au gândit niciodată nerespectuos, poate chiar "urât" la adresa profesorilor? Presupun că vă întrebați: În general sau în particular? Am ajuns la concluzia, după mulți ani de "școală" că problema fundamentală, pentru elev, constă tocmai în lipsa capacității acestuia de a diferenția între *general* și *particular*. În liceu am avut profesori la care am ținut, alții pe care i-am respectat și alții care au avut efectul cel mai ciudat asupra mea: am absolvit liceul și am jurat că mă voi face orice, numai profesor nu!

De ce nu am vrut să fiu profesor? Pentru că nu aveam nici o garanție că nu voi deveni o "fosilă" de care râd toți copiii, că nu mă voi deprofesionaliza, că nu voi deveni rigidă, agresivă și ridicolă. Azi doar sper că nu sunt..., și, evident mă străduiesc permanent să devin un profesor care să difere (în general și în particular) de acel profesor care m-a speriat în anii de școlar. Regretabil, dar unii mă sperie chiar și acum pentru că privesc elevul ca pe un obiect, caută cu înverșunare să-l "prindă", să-l "încuie" cu ceea ce acesta *nu știe*. Acești profesori nici nu încearcă să pună în valoare *ceea ce știe* elevul. Sunt destui elevi și studenți care, datorită experiențelor dezastruoase trăite în timpul studiilor, își etichetează toți profesorii ca fiind "dușmani". O generalizare care doare... Și culmea! Îl doare pe acel profesor care nu ar merita eticheta, pe când, evident, celui care a provocat acest sentiment de mult nu-i mai pasă ce gândesc elevii sau studenții.

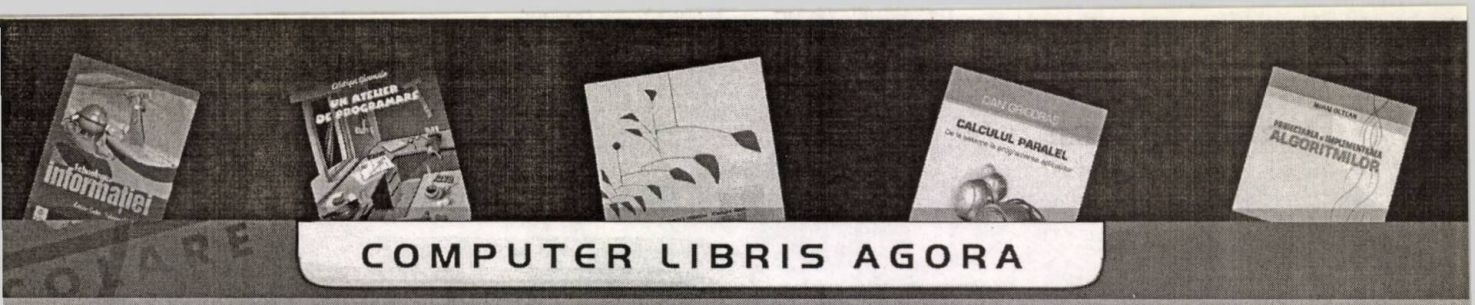
Stau și mă gândesc, cum este posibil că unii colegi de-ai noștri nu înțeleg câteva lucruri esențiale. Cum este posibil să uite irevocabil propriile experiențe? Cum nu-și aduc aminte pe cine și de ce au respectat sau, dimpotrivă, au judecat cu asprimea caracteristică tinereții? Dar anii trec, generațiile de profesori se schimbă și problemele rămân, mai bine zis se regenerează, se perpetuează. Și bineînțeles, de data aceasta nu meditez asupra unor întâmplări, ci asupra unui fenomen. Nu este vorba de profesorul sever, pretențios care la rândul lui este pretențios și cu el însuși. Nu este vorba nici de acel profesor, aflat la început de carieră care încă intră în clasă cu emoții și cu oarecare frică de elevi, mai bine zis de întrebările lor. Este vorba de cel care "nu dă 10" pentru că nota 10 este "a lui"..., despre cel care îi jignește de fiecare dată când ei nu știu răspunsul corect, uitând că acești copii vin la școală să învețe și că tocmai el/ea ar trebui să-i învețe. Și nu l-am amintit pe cel care dă note mari, nemeritate, crezând că astfel va ascunde tot ceea ce nu a predat, nu a explicat.

În multe școli, unele clase de elevi sunt foarte repede categorisiți drept proști, slabi, leneși etc. Dar chiar sunt curioasă stimați cititori: cum i-ați descrie pe copiii *deștepți*? Să fie cei care învață pe linia de plutire și în loc să tocească tot ce prevede programa, se mai și distrează? Sau cei care învață pentru cunoștințe dar și pentru note, și țin minte că o anumită noțiune pentru profesorul *x* trebuie redată într-un fel, și pentru *y* altfel. Sau cei care se duc la olimpiade? Ce se întâmplă acolo? Li se demonstrează că de fapt și ei sunt *proști*... În clasa lor, în școala lor au crezut că se pricep să rezolve probleme, dar la olimpiada națională nu mai reușesc să facă nimic. Cel puțin așa reiese din rezultate. De exemplu, la ONI 2002 punctajul mediu la clasele 11-12 a fost 60 de puncte din 600 posibile! Subiect de meditație... Oare, dintr-o dată copiii au devenit mai "slabi", sau de vină să fie programa care prevede mai puține ore, deci elevii de azi primesc mai puține cunoștințe decât cei de acum 5-10 ani? Sunt întrebări retorice. Aceste olimpiade ar trebui să fie prilej de bucurie pentru mulți copii, ar trebui să li să dea ocazia să arate ce știu. Vorba lui *Bogdan Dumitru* - medaliat cu aur la IOI și cu argint la CEOI în 1999 etc. - "*păi ăștia au fumat un cartuș de țigări până au inventat enunțul ăsta și se așteaptă ca noi să rezolvăm în patru ore trei monștri de probleme!*"

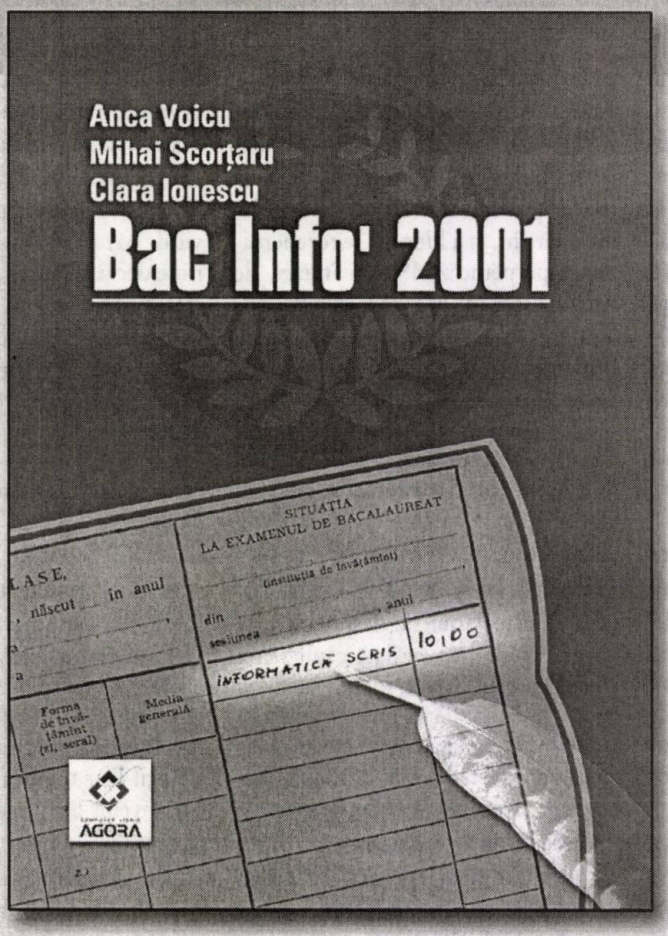
Și dacă ar suferi doar olimpicii... Dar suferă toți ceilalți cu care la ore profesorul nu s-a ocupat deloc pentru că pompa cunoștințe în doi-trei copii mai "răsăriți", neglijându-i complet pe ceilalți. Dar tot efortul, tot sacrificiul s-a dovedit a fi degeaba, pentru că la concurs li s-au pus în față probleme pe care nici profesorii nu le pot rezolva. "Slavă cerului, nici nu trebuie!", va zice cineva.

Rămân la părerea mea că olimpiadele ar trebui să fie serbarea reușitei, a succesului, a mulțumirii că nu s-a muncit inutil. Ar trebui să se deruleze într-o efervescență care să fie încununarea unei munci depuse cu pasiune de niște *COPII* care nu sunt neapărat genii. Dar în condițiile în care problemele sunt *IMPOSIBIL* de rezolvat în timpul acordat, în condițiile în care sunt necesare cunoștințe care numai cu sacrificii deosebite s-ar fi putut acumula, rămâne tristețea că îi vom pierde și pe cei pentru care performanța ar fi fost posibilă.

Clara Ionescu



COMPUTER LIBRIS AGORA



Bac Info' 2001 conține 20 de variante de subiecte pentru Bacalaureat. Fiecare variantă conține câte patru subiecte. Primul dintre ele conține trei subpuncte care testează capacitățile candidaților de a recunoaște valoarea de adevăr a unei propoziții, de a determina efectul unui anumit algoritm și de a găsi o anumită greșeală într-un algoritm. Al doilea subiect conține doua subpuncte; primul testează deprinderea candidaților de a folosi limbajul pseudocod, iar al doilea cea de a folosi un limbaj de programare (Pascal sauC++). Al treilea subiect este o problemă pentru rezolvarea căreia este necesară folosirea unor structuri dinamice de date, iar al patrulea este o problemă pentru a cărei rezolvare trebuie folosită metoda backtracking.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

AMINTIRI

Probabil puțini dintre cititori își amintesc de vechea Gazetă de Informatică. Era o revistă care conținea foarte multe probleme propuse, cititorii trimiteau o mulțime de soluții care erau, uneori, publicate. Evident, problemele de atunci nu respectă standardele actuale; vă prezentăm acum câteva enunțuri de la mijlocul anilor '90 "transformate" astfel încât să poată fi folosite în mileniul III.

P070201: Influențe

Olimpiada Județeană de Informatică, clasa a IX-a, Botoșani 1994

Se consideră un grup de N persoane și o mulțime de perechi (i, j) cu semnificația *persoana i este influențată de persoana j*. Două persoane nu se pot influența reciproc în mod direct. Va trebui să verificați dacă există persoane care se pot influența reciproc prin intermediul altor persoane.

Influența unei persoane este dată de numărul persoanelor care sunt influențate, direct sau indirect, de persoana respectivă.

Un *lanț al slăbiciunilor* este un șir de persoane $a_1, a_2, \ldots, a_n$ astfel încât persoana a_i influențează persoana a_{i+1} ($i = 1, \ldots, n - 1$).

În cazul în care nu există persoane care se pot influența reciproc, va trebui să determinați persoanele cu cea mai mare influență, precum și unul dintre cele mai lungi lanțuri ale slăbiciunilor.

Date de intrare

Fișierul de intrare `PERSOANE.IN` va conține pe prima linie numărul N al persoanelor și numărul M al perechilor. Fiecare dintre următoarele M linii va conține două numere i și j, cuprinse între 1 și N, cu semnificația *persoana i este influențată de persoana j*.

Date de ieșire

În cazul în care există persoane care se pot influența reciproc, prima linie a fișierului de ieșire `PERSOANE.OUT` va conține șirul DA. În caz contrar, pe această linie se va scrie șirul NU, pe a doua linie se vor scrie numerele de ordine ale celor mai influente persoane, iar pe a treia linie se vor scrie numerele de ordine ale persoanelor din cel mai lung lanț al slăbiciunilor, începând cu cea mai influentă persoană din lanț și terminând cu cea mai puțin influentă. Numerele care se află pe aceeași linie în fișierul de ieșire vor fi separate prin spații.

Restricții

- $2 \le N \le 100$;
- $1 \le M \le N \cdot (N - 1)/2$.

Exemple

`PERSOANE.IN`

```
3 3
1 2
2 3
3 1
```

`PERSOANE.OUT`

```
DA
```

`PERSOANE.IN`

```
23 21
1 2
2 3
3 4
1 5
5 6
6 7
1 8
8 9
10 11
11 12
10 13
13 14
14 15
15 16
10 17
17 18
19 20
20 21
21 22
22 23
23 15
```

PERSOANE.OUT

```
NU
1 10
19 20 21 22 23 15 16
```

Timp de execuție: 1 secundă/test

P070202: Păsărească

Olimpiada Județeană de Informatică, clasa a IX-a, Cluj 1994

Pentru a transforma un cuvânt din limba română în limba păsărească, fiecare vocală *v* este înlocuită cu grupul de trei litere în care prima și ultima este vocala *v*, iar a doua este litera `'p'`. Va trebui să scrieți un program care realizează "traduceri" din limba română în limba păsărească și invers.

Date de intrare

Fișierul de intrare **CUVINTE.IN** conține pe prima linie numărul 1 sau numărul 2. Dacă pe prima linie se află numărul 1, atunci pe a doua se vor afla cuvinte din limba română, formate din litere mici ale alfabetului englezesc și separate prin spații. Dacă numărul de pe prima linie este 2, atunci cuvintele de pe a doua linie vor fi scrise în limba păsărească.

Date de ieșire

Prima linie a fișierului de ieșire **CUVINTE.OUT** va conține cuvintele traduse (din română în păsărească sau invers), separate prin câte un spațiu, în ordinea în care apar acestea în fișierul de intrare.

Precizări

- Numărul cuvintelor este cuprins între 1 și 1000.
- Un cuvânt va fi format din cel mult 1000 de litere.

Exemple

CUVINTE.IN

```
1
gazeta de informatica
```

CUVINTE.OUT

```
gapazepetapa depe ipinfopormapatipicapa
```

CUVINTE.IN

```
2
gipinfopo
```

CUVINTE.OUT

```
ginfo
```

Timp de execuție: 1 secundă/test

P070203: Piramida

Olimpiada Județeană de Informatică, clasa a IX-a, Cluj 1994

Se consideră următoarea "piramidă" de numere prezentată în continuare:

```
                1
              1 1
              2 1
            1 2 1 1
          1 1 1 2 2 1
          3 1 2 2 1 1
        1 3 1 1 2 2 2 1
        1 3 2 1 3 2 1 1
```

Pe prima linie se află doar numărul 1; fiecare linie se obține din linia anterioară prin indicarea perechilor de forma *factor de repetiție*, număr. De exemplu, din linia 1 1 1 2 2 1 se obține linia 3 1 2 2 1 1 deoarece pe primele poziții apare de trei ori numărul 1, apoi de două ori numărul 2 și o dată numărul 1. Folosind piramida se poate construi un șir infinit care conține liniile piramidei, începând cu prima. Așadar, primele elemente ale șirului sunt 1, 1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 2, 2, 1, 3, 1, 2, 2, 1, 1, ... Se poate demonstra matematic faptul că piramida va conține doar numere cuprinse între 1 și 3.

Dându-se două numere naturale p și q ($p < q$) se cere să se determine valoarea zecimală a numărului a cărui cifre, în baza 4, sunt cele de pe pozițiile $p, p + 1, \ldots, q - 1, q$ (în această ordine) ale șirului obținut din piramidă. Se consideră că primul număr din șir se află pe poziția 0.

Date de intrare

Fișierul de intrare **PIRAMIDA.IN** va conține o singură linie pe care se vor afla numerele p și q, separate printr-un singur spațiu.

Date de ieșire

Fișierul de ieșire **PIRAMIDA.OUT** va conține numărul a cărui cifre, în baza 4, sunt cele de pe pozițiile $p, p + 1, \ldots, q - 1, q$ (în această ordine) ale șirului obținut din piramidă.

Restricții

- $0 < p < q < 10000$;
- $q - p < 16$.

Exemplu

PIRAMIDA.IN

```
4 7
```

PIRAMIDA.OUT

```
89
```

Timp de execuție: 1 secundă/test

P070204: Numere

Olimpiada Județeană de Informatică, clasa a IX-a, Cluj 1994

Pentru un număr natural scris sub forma $c_1c_2\ldots c_n$ se definește oglinditul său ca fiind numărul obținut prin citirea cifrelor în ordine inversă ($c_n\ldots c_2c_1$). De exemplu, oglinditul numărului 127 este 721.

Se spune că un număr are *aspect de munte* dacă există un număr natural k ($1 < k < n$) astfel încât $c_1 < c_2 < \ldots < c_k$ și $c_k > c_{k+1} > \ldots > c_n$. De exemplu, numărul 28731 are *aspect de munte* ($k = 2$). Analog, un număr are *aspect de vale* dacă există un număr natural k ($1 < k < n$) astfel încât $c_1 > c_2 > \ldots > c_k$ și $c_k < c_{k+1} < \ldots < c_n$. De exemplu, numărul 97635 are *aspect de vale* ($k = 4$).

Fiind date două numere naturale a și b ($a < b$) să se determine o pereche de numere din intervalul $[a, b]$ care au același aspect (ambele munte sau ambele vale) cu proprietatea că oglinditul sumei numerelor este egal cu suma oglinditelor numerelor.

Date de intrare

Fișierul de intrare **NUMERE.IN** conține o singură linie pe care se află valorile a și b, separate printr-un spațiu.

Date de ieșire

Fișierul de ieșire **NUMERE.OUT** va conține o singură linie pe care se vor afla două numere care respectă condițiile impuse.

Restricție

- $100 < a < b < 10000$.

Exemplu

NUMERE.IN

```
1200 1600
```

NUMERE.OUT

```
1231 1532
```

Timp de execuție: 1 secundă/test

P070205: Cel mai mare divizor comun

Olimpiada Județeană de Informatică, clasa a IX-a, Sălaj 1994

Se consideră un șir de n numere naturale cuprinse între 2 și 2000000000. Să se determine cel mai mare divizor comun al celor n numere și, știind că acesta este întotdeauna mai mare decât 1, să se determine toți divizorii săi primi.

Date de intrare

Prima linie a fișierului de intrare **CMMDC.IN** va conține numărul n al elementelor din șir. Următoarea linie va conține cele n elemente ale șirului, separate prin spații.

Date de ieșire

Prima linie a fișierului de ieșire **CMMDC.OUT** va conține cel mai mare divizor comun d al elementelor din șir. Pe cea de-a doua linie se vor afla divizorii primi ai numărului d, separați prin câte un spațiu.

Exemple

CMMDC.IN

```
4
630 126 567 315
```

CMMDC.OUT

```
63
3 7
```

CMMDC.IN

```
3
12 24 36
```

CMMDC.OUT

```
12
2 3
```

Timp de execuție: 1 secundă/test

P070206: Numere deosebite

Olimpiada Județeană de Informatică, clasa a IX-a, Teleorman 1994

Se spune că un număr n este deosebit dacă există un număr m astfel încât $m + S(m) = n$, unde $S(m)$ este suma cifrelor numărului m.

Dându-se un șir de numere, să se verifice care dintre ele sunt deosebite și care nu.

Date de intrare

Prima linie a fișierului de intrare **DEOSEBIT.IN** va conține numărul k al elementelor din șir. Fiecare dintre următoarele k linii va conține câte un număr natural.

Date de intrare

Pentru fiecare element al șirului, în fișierul de ieșire **DEOSEBIT.OUT** se va scrie cuvântul DA pe linie, dacă numărul corespunzător din fișierul de intrare este deosebit și NU în caz contrar.

Restricții

- $1 \le k \le 1000$;
- elementele șirului sunt numere naturale cuprinse între 1 și 1000.

Exemplu

DEOSEBIT.IN

```
3
1
2
3
4
5
```

DEOSEBIT.OUT

```
NU
DA
NU
DA
NU
```

Timp de execuție: 1 secundă/test

Olimpiada NAȚIONALĂ de Informatică

Vă prezentăm în continuare soluțiile problemelor propuse la ONI 2002 la clasele a XI-a și a XII-a. Aceste soluții au fost realizate de redacția GInfo pe baza soluțiilor oficiale prezentate de autorii problemelor și au fost verificate folosind seturile de date cu care au fost testate soluțiile concurenților.

P050213: Arbore

Vom spune că un nod este ***rezolvat*** dacă și numai dacă el *face parte din exact un ciclu*. Similar, vom spune că un subarbore este ***rezolvat*** dacă și numai dacă *toate nodurile acestuia sunt rezolvate*.

Vom numi ***fir*** un subarbore care este *lanț* (fiecare nod, cu excepția frunzei, are un singur fiu). Un fir este ilustrat în figura 1.

Figura 1: Fir

Se observă că un fir poate fi rezolvat foarte simplu prin *unirea rădăcinii sale cu frunza sa*. Evident, firul va putea fi rezolvat numai dacă el *conține cel puțin două noduri*. Procedeul este ilustrat în figura 2.

Figura 2: Fir rezolvat

Pentru a rezolva această problemă vom alege, pentru început, un nod care va fi rădăcina arborelui. Vom spune că un subarbore este ***potențial rezolvabil*** dacă *toți fiii rădăcinii subarborelui* (în număr de cel puțin doi) *sunt rădăcini ale unor fire*.

În continuare vom arăta modul în care poate fi rezolvat un nod potențial rezolvabil. Vom alege cele mai scurte două fire (acestea pot conține unul, două sau mai multe noduri) și *vom uni printr-o muchie frunzele acestor fire*. Astfel vom obține un ciclu care va conține nodurile din componența celor două fire, precum și nodul potențial rezolvabil. Eventualele fire rămase (dacă sunt formate din cel puțin trei noduri) vor fi rezolvate așa cum se arată în figura 2. Se observă că, în cazul în care există mai mult de două fire care conțin unul sau două noduri, subarborele nu poate fi rezolvat și problema *nu are soluție*. Rezolvarea unui subarbore potențial rezolvabil este ilustrată în figura 3.

După rezolvarea unui subarbore, acesta este ***eliminat*** din arbore și se va alege un alt nod potențial rezolvabil. Procedeul continuă până la *rezolvarea tuturor nodurilor* sau până la *detectarea unui subarbore care nu poate fi rezolvat*.

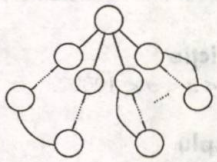


Figura 3: Rezolvarea unui subarbore potențial rezolvabil

În cele ce urmează vom prezenta modul în care poate fi implementat algoritmul descris. Pentru început vom alege o modalitate de reprezentare a arborelui. Vom construi o *listă de fii* și vom păstra, pentru fiecare nod, *indicele primului și al ultimului fiu* în această listă. Prin convenție vom considera că pentru un nod care nu are nici un fiu (frunză) indicele ultimului fiu va fi mai mic decât indicele primului fiu; de asemenea, pentru fiecare nod vom păstra *nodul părinte*. Datorită faptului că arborele este dat prin șirul muchiilor, va trebui să îl construim. Pentru aceasta vom realiza o parcurgere în lățime a arborelui, ceea ce permite construirea listei fiilor pe măsura traversării nodurilor arborelui.

În continuare vom alege rădăcina arborelui; aceasta poate fi orice nod care are *gradul mai mare decât 1*. Vom continua cu identificarea fiilor; evident, toate frunzele sunt fire de lungime 1. Pentru fiecare frunză vom "urca în arbore" atâta timp cât nodul curent are un singur fiu (părintele frunzei va fi rădăcina unui fir de lungime 2, părintele părintelui va fi rădăcina unui fir de lungime 3 și așa mai departe). Folosind acest procedeu vom determina și lungimile firelor. Prin convenție, vom considera că dacă un nod nu face parte dintr-un fir, atunci lungimea corespunzătoare va fi -1.

Acum, la fiecare pas vom căuta un nod care conține cel puțin doi fii nerezolvați care sunt rădăcini ale unor fire. Vom determina numărul firelor de lungime 1 sau 2 și, în cazul în care există cel mult două astfel de fire, vom rezolva subarborele așa cum se arată în figura 3. Dacă există mai mult de două astfel de fire vom scrie în fișierul de ieșire valoarea -1 și vom opri execuția programului. După rezolvarea subarborelui, acesta este eliminat din arbore prin mar-

carea tuturor nodurilor sale ca fiind rezolvate. Acum părintele rădăcinii subarborelui ar putea să facă parte dintr-un fir. În cazul în care părintele a rămas cu un singur fiu care este rădăcină a unui fir, atunci părintele devine rădăcină a unui fir de lungime cu 1 mai mare. Vom "urca" din nou în arbore până la întâlnirea unui nod care are mai mulți fii nerezolvați și vom păstra, pentru fiecare nod, lungimea firului cu rădăcina în nodul respectiv. O a doua posibilitate este ca părintele rădăcinii subarborelui eliminat să devină frunză. Se va "urca" în arbore în mod asemănător, dar noua frunză va fi rădăcina unui fir de lungime 1.

Există posibilitatea ca, la un moment dat, să rămânem cu un arbore format dintr-un singur fir. Dacă acesta are lungimea mai mare decât 2, atunci el va fi rezolvat așa cum se arată în figura 2. În caz contrar, problema nu are soluție.

De fiecare dată când rezolvăm un subarbore, vom păstra o listă cu muchiile adăugate. În final, vom scrie în fișierul de ieșire numărul muchiilor adăugate, precum și extremitățile muchiilor din listă.

Analiza complexității

Citirea datelor de intrare se realizează în timp **liniar**, deoarece implică citirea extremităților celor N - 1 muchii ale arborelui.

Construirea arborelui prin parcurgerea în lățime implică, pentru fiecare nod parcurs, studierea tuturor muchiilor. Deoarece avem N noduri și N - 1 muchii, ordinul de complexitate al operației este $O(N^2)$. Pe parcursul construirii arborelui se creează șirul părinților, cel al fiilor, cel al gradelor, precum și cele care păstrează indicii primului și ultimului fiu pentru fiecare nod.

Pentru păstrarea informațiilor inițiale despre fire, vom parcurge, în cel mai defavorabil caz, toate nodurile arborelui, deci operația se realizează în timp **liniar**. Trebuie observat faptul că acest ordin de complexitate include și timpul consumat pentru actualizarea necesară după eliminarea unui nod din arbore.

Alegerea unui nod potențial rezolvabil implică traversarea listei fiilor pentru a verifica dacă sunt rădăcini ale unor fire, deci ordinul său de complexitate este $O(N)$. La fiecare pas vom elimina cel puțin trei noduri, deci vom realiza cel mult $N / 3$ astfel de căutări. Ca urmare, ordinul de complexitate al tuturor alegerilor este $O(N^2)$.

Pentru rezolvarea arborelui este necesară parcurgerea tuturor nodurilor acestuia. Un nod rezolvat nu va fi parcurs încă o dată. Așadar, ordinul de complexitate al acestei operații este $O(N)$.

Ordinul de complexitate al operațiilor de eliminare a nodurilor este tot $O(N)$ pentru că fiecare nod este eliminat o singură dată.

Scrierea datelor în fișierul de ieșire se realizează tot în timp **liniar** pentru că implică scrierea extremităților a cel mult $N / 3$ muchii.

În concluzie, algoritmul de rezolvare a acestei probleme are ordinul de complexitate $O(N) + O(N^2) + O(N) + O(N^2) + O(N) + O(N) + O(N) = O(N^2)$.

Dificultate 57

Algoritmi	★★★☆☆
Structuri de date	★★☆☆☆
Originalitate	★★☆☆☆
Cunoștințe	★★★☆☆
Implementare	★★★☆☆

www.ginfo.ro/revista/12_7/surse/arbore.cpp

P050214: Decodificare

Inițial vom determina numărul de poziții în care permutarea identică și permutarea care trebuie determinată coincid.

În continuare vom încerca să determinăm pozițiile în permutare ale tuturor numerelor cuprinse între 1 și N. Pentru fiecare număr i, dacă poziția sa nu a fost determinată la un pas anterior, va trebui să determinăm poziția acestuia în permutarea căutată. Pentru aceasta, în permutarea identică, vom interschimba, pe rând, numărul i cu fiecare dintre celelalte elemente. Astfel, vom obține permutări de forma $(1, 2, 3, ..., i - 1, j, i + 1, ..., j - 1, i, j + 1, ..., N)$. Numărul pozițiilor, în care această permutare și permutarea care trebuie considerată coincid, poate crește dacă și numai dacă numărul i ajunge pe poziția sa sau numărul care trebuie să se afle pe poziția i ajunge în această poziție. Așadar, pentru doar două dintre aceste permutări numărul pozițiilor considerate poate să crească. Există trei cazuri:

- Numărul pozițiilor care coincid nu crește în nici un caz. În această situație putem trage concluzia că numărul i se află deja pe poziția sa, deci va fi al i-lea element în permutarea care trebuie determinată.
- Numărul pozițiilor care coincid crește o singură dată. Fie j numărul cu care a fost interschimbat i în permutarea care a dus la creșterea numărului de poziții care coincid. Este ușor de observat că, în permutarea care trebuie determinată, numărul i se va afla pe poziția j, iar numărul j se va afla pe poziția i.
- Numărul pozițiilor care coincid crește de două ori. Fie j și k pozițiile pe care a ajuns numărul i în cele două permutări. Va trebui să determinăm pe care dintre aceste două poziții se află i în permutarea care trebuie determinată. Știm cu siguranță că unul dintre numerele j și k se află pe poziția i în permutarea care trebuie determinată. Va trebui să studiem două cazuri:
  - i trece pe poziția j, j pe poziția k și k pe poziția i;
  - i trece pe poziția k, j pe poziția i și k pe poziția j.

  În unul dintre aceste două cazuri numărul pozițiilor care coincid va fi cu cel puțin 2 mai mare decât numărul pozițiilor care coincid în permutarea identică. Așadar, vom verifica una dintre permutări; dacă numărul pozițiilor care coincid crește cu cel puțin 2, atunci vom ști care este poziția numărului i în permutarea care trebuie determinată și care este numărul care se află pe poziția i în această permutare.

După determinarea pozițiilor tuturor numerelor, vom afișa permutarea determinată.

Analiza complexității

Pentru început, trebuie precizat faptul că apelul funcției `Check` implică traversarea unui vector cu N elemente care conține permutarea care trebuie determinată. Așadar ordinul de complexitate al unui astfel de apel este $O(N)$.

Preluarea dimensiunii șirului se realizează în timp constant, deci are ordinul de complexitate $O(1)$.

Identificarea poziției unui număr implică realizarea a cel mult $N - 1$ interschimbări și, eventual, a unei permutări care va diferi de permutarea identică în trei poziții. Pentru fiecare dintre aceste permutări se va realiza un apel al funcției `Check`, deci operația de determinare a poziției unui număr în permutarea căutată are ordinul de complexitate $O(N) \cdot O(N) = O(N^2)$. În total, vom efectua cel mult N determinări, deci ordinul de complexitate al operației de determinare a permutării căutate este $O(N) \cdot O(N^2) = O(N^3)$.

Scrierea datelor în fișierul de ieșire implică traversarea permutării determinate, deci ordinul de complexitate al operației este $O(N)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(1) + O(N^3) + O(N) = O(N^3)$.

Dificultate 27

Algoritmi	★☆☆☆☆
Structuri de date	★☆☆☆☆
Originalitate	★★☆☆☆
Cunoștințe	★★★★☆
Implementare	★★☆☆☆

www.ginfo.ro/revista/12_7/surse/decod.cpp

P050215: SETI

Pentru a determina numărul de apariții ale cuvintelor din dicționar în mesajul dat, vom ordona alfabetic toate șirurile (cuvintele) de 16 caractere consecutive din mesaj. Apoi, pentru fiecare cuvânt din dicționar vom determina, prin metoda căutării binare, prima și ultima apariție a cuvântului în șirul sortat. Evident, dacă un cuvânt din dicționar conține k litere, atunci, în momentul în care determinăm aceste două poziții, vom lua în considerare doar primele k litere ale șirurilor.

O primă observație ar fi că, datorită faptului că nu avem suficientă memorie la dispoziție, șirul sortat trebuie să conțină indici și nu cuvinte propriu-zise. Astfel, al i-lea element al șirului sortat va conține poziția la care începe al i-lea cuvânt în șirul sortat.

Datorită limitei de timp foarte stricte, algoritmii clasici de sortare (*quicksort*, *heapsort*) nu sunt suficient de performanți. Pentru această problemă este recomandabilă folosirea algoritmului *radixsort* (vezi *CLR*[1], p. 152).

O altă dificultate care apare este datorată tot insuficienței memoriei. Șirul sortat de indici ar trebui să conțină numere reprezentate pe patru octeți (teoretic ar fi suficienți trei) deoarece textul poate conține peste 100000 de cuvinte. Pentru a evita această problemă, vom împărți textul în patru blocuri și vom aplica algoritmul descris pentru fiecare dintre ele. Fiecare bloc va conține cel mult $2^{15} = 32768$ litere. Datorită faptului că numărul de litere este cel mult $2047 \cdot 64 = 131008 < 2^{17} = 131072$, sunt suficiente patru blocuri. Vom avea și patru șiruri sortate, dar fiecare element al acestor șiruri va putea fi reprezentat pe doi octeți.

Datorită împărțirii textelor în blocuri apar probleme pentru cuvintele care fac parte din două blocuri. Pentru a le evita, vom face în așa fel încât ultimele 15 litere dintr-un bloc să fie adăugate ca primele 15 litere ale blocului următor. Dimensiunea totală a textului va crește cu până la 45 de caractere, dar va ajunge doar la cel mult $131053 < 2^{17}$, deci nu apar blocuri suplimentare. Se observă că la începutul celui de-al doilea bloc sunt copiate 15 litere din primul, deci ultimele 15 litere ale acestui al doilea grup trebuie să fie mutate în al treilea. Alte 15 litere ajung să fie ultimele în al doilea bloc, deci trebuie copiate în al treilea. Așadar, la începutul celui de-al treilea bloc vor fi ultimele 30 de litere ale celui de-al doilea bloc. Folosind același raționament, ajungem la concluzia că la începutul celui de-al patrulea bloc vor fi ultimele 45 de litere ale celui de-al treilea bloc.

Așadar, pentru rezolvarea problemei, mai întâi vom împărți textul inițial în grupuri de 32768 de litere. Apoi vom realiza copierea grupurilor de 15, 30 și 45 de litere dintr-un bloc în altul și vom construi cele patru șiruri sortate, folosind algoritmul *radixsort*. În continuare, pentru fiecare cuvânt din dicționar, vom determina prima și ultima apariție a sa în fiecare dintre șirurile sortate (dacă nu există nici o apariție vom spune, prin convenție, că poziția ultimei apariții este mai mică decât poziția primei apariții). Numărul de cuvinte dintre cele două poziții va fi adăugat la numărul de apariții în text ale cuvântului din dicționar. Pe măsură ce sunt determinate aparițiile cuvintelor din dicționar, se vor scrie în fișierul de ieșire numerele corespunzătoare.

Analiza complexității

Citirea textului care trebuie analizat se realizează linie cu linie, deci ordinul de complexitate al operației este $O(N)$. Deoarece dicționarul conține M cuvinte, ordinul de complexitate al operației de citire a acestor cuvinte este $O(M)$.

Construirea blocurilor se realizează pe parcursul citirii, ordinul de complexitate al operației fiind $O(N)$. Aceeași complexitate ($O(N)$) o are și operația de transformare a blocurilor astfel încât ultimele 15 litere dintr-un grup să fie identice cu primele 15 din următorul.

Algoritmul *radixsort* este *liniar*, deci ordinul de complexitate al operației de sortare este $O(N)$.

Pentru fiecare cuvânt din dicționar vom efectua cel mult opt căutări binare, ordinul de complexitate al unei căutări fiind $O(\log N)$. Pentru cele 8 căutări vom avea ordinul de complexitate $8 \cdot O(\log N) = O(\log N)$. Deoare-

1. *Cormen T. H., Leiserson C. E., Rivest R. L., Introducere în Algortmi, Computer Libris Agora, Cluj-Napoca, 2000*

ce pentru fiecare dintre cele M cuvinte din dicționar se vor realiza căutări binare, ordinul de complexitate al operației de determinare a numărului de apariții pentru toate cuvintele din dicționar este $O(M) \cdot O(\log N) = O(M \cdot \log N)$.

Generarea datelor de ieșire implică scrierea a M numere, deci ordinul de complexitate al operației este $O(M)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(N) + O(M) + O(N) + O(N) + O(N) + O(M \cdot \log N) + O(M) = O(M \cdot \log N + N)$.

Dificultate **51**

Algoritmi	★★☆☆☆
Structuri de date	★★★☆☆
Originalitate	★★★☆☆
Cunoștințe	★★★★☆
Implementare	★★☆☆☆

www.ginfo.ro/revista/12_7/surse/seti.cpp

P050216: Suma divizorilor

Numărul A poate fi scris ca produs de factori primi sub forma:

$$A = d_1^{p_1} \cdot d_2^{p_2} \cdot \text{K} \cdot d_n^{p_n},$$

unde $d_1, ..., d_n$ sunt divizorii primi ai lui A, iar $p_1, ..., p_n$ sunt puterile la care apar aceștia în descompunerea lui A în factori primi.

Ca urmare, numărul A^B poate fi scris sub forma:

$$A^B = d_1^{p_1 \cdot B} \cdot d_2^{p_2 \cdot B} \cdot \text{K} \cdot d_n^{p_n \cdot B}$$

Așadar, divizorii numărului A^B au forma $\prod_{i=1}^{n} d_i^{q_i}$, unde q_i variază între 0 și $p_i \cdot B$.

Suma acestor divizori poate fi scrisă sub forma:

$$\sum_{\substack{q_i=0,\text{K},p_i \cdot B \\ i=1,\text{K},n}} \prod_{i=1}^{n} d_i^{q_i}.$$

Această sumă de produse poate fi scrisă sub forma unui produs de sume astfel:

$$\prod_{i=1}^{n} \sum_{j=0}^{p_i \cdot B} d_i^j.$$

Așadar, suma divizorilor unui număr de forma A^B este dată de un produs de sume. Aceste sume au forma $1 + q + q^2 + q^3 + ... + q^m$, deci sunt sume ale unei progresii geometrice de rație q. Este cunoscut faptul că valoarea unei astfel de sume, pentru $q > 1$ este $\frac{q^{m+1} - 1}{q - 1}$, iar pentru $q = 1$ este $m + 1$.

Datorită faptului că se cere determinarea restului împărțirii la 9901 a sumei divizorilor, toate operațiile efectuate vor fi *modulo* 9901.

Vor trebui folosite relațiile:

`(a + b) mod n = ((a mod n) + (b mod n)) mod n` și

`(a · b) mod n = ((a mod n) · (b mod n)) mod n`.

De asemenea, datorită faptului că numărul 9901 este prim, pentru orice număr nenul a va exista un număr a^{-1} (numit *invers*) astfel încât $(a \cdot a^{-1} \bmod 9901 = 1)$. Așadar, pentru a efectua o împărțire cu un anumit număr vom realiza, de fapt, o înmulțire cu inversul acestuia.

Pentru a rezolva problema vom începe cu descompunerea numărului A în factori primi și identificarea puterilor la care apar aceștia în descompunere. Vom înmulți aceste puteri cu B și apoi vom calcula sumele progresiilor corespunzătoare.

Pentru calculul sumei unei progresii avem nevoie de efectuarea unei ridicări la putere (evident, *modulo* 9901). O modalitate rapidă de calcul se bazează pe următoarea formulă recursivă:

$$\begin{cases} 1 & \text{pentru } n = 0 \\ a^{[n/2]} \cdot a^{[n/2]} & \text{pentru } n \neq 0 \text{ par} \\ a^{[n/2]} \cdot a^{[n/2]} \cdot a & \text{pentru } n \neq 0 \text{ impar} \end{cases}$$

O modalitate de determinare a inversului unui număr a (*modulo* 9901) este considerarea numerelor de forma $9901 \cdot k + 1$ și alegerea primului număr b de această formă al cărui rest la împărțirea cu a este 1. Inversul va fi dat de valoarea b / a. Datorită faptului că 9901 este prim va exista întotdeauna un astfel de număr și se poate demonstra că valoarea k corespunzătoare este mai mică decât 9901.

O altă observație necesară este faptul că scăderea valorii 1 (necesară pentru calcularea sumei progresiilor) este echivalentă cu adunarea valorii 9900 (*modulo* 9901).

Pe parcursul determinării sumei progresiilor, acestea vor fi înmulțite, *modulo* 9901. În final, rezultatul va fi scris în fișierul de ieșire.

Analiza complexității

Datele de intrare constau doar în două numere, deci operația de citire a acestora se realizează în timp *constant*.

Pentru determinarea divizorilor primi și a puterilor la care apar aceștia în descompunerea numărului A vom lua în considerare numărul 2 și numerele impare cuprinse între 3 și cel mult $\sqrt{A}$, așadar ordinul de complexitate al acestei operații este $O(\sqrt{A})$.

Numărul sumelor calculate este dat de numărul divizorilor primi ai numărului A. Folosind aproximarea lui *Stirling* se poate deduce că există aproximativ $\ln A$ numere prime cuprinse între 2 și A. Așadar, vom calcula $O(\log A)$ sume. Calculul unei sume implică o ridicare la putere. Folosind algoritmul recursiv prezentat anterior, o ridicare la puterea n are ordinul de complexitate $O(\log n)$. Puterea la care poate apărea un factor prim în descompunerea numărului A este cel mult $\log_2 A$; în descompunerea lui A^B puterea poate fi cel mult $B \cdot \log_2 A$. Ca urmare, o ridicare la putere va avea ordinul de complexitate $O(\log (B \cdot \log A))$. Celelalte operații necesare pentru calculul sumelor se realizează în timp constant (considerăm că determinarea inversului are ordinul de complexitate $O(1)$ deoarece operația necesită cel mult 9901 pași; există și posibilitatea construirii și utilizării unui tablou de constante a, unde a_i să conțină inversul numărului i modulo 9901), deci ordinul de

complexitate al determinării sumei tuturor divizorilor este $O(\log A) \cdot O(\log (B \cdot \log A)) = O(\log A \cdot \log (B \cdot \log A))$.

În fișierul de ieșire trebuie scris un singur număr, deci ordinul de complexitate al operației de scriere a datelor de ieșire este $O(1)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(1) + O(\sqrt{A}) + O(\log A \cdot \log (B \cdot \log A)) + O(1) = O(\sqrt{A} + \log A \cdot \log(B \cdot \log A))$.

Dificultate **37**

Algoritmi	★☆☆☆☆
Structuri de date	☆☆☆☆☆
Originalitate	★★☆☆☆
Cunoștințe	★★★★★
Implementare	★☆☆☆☆

www.ginfo.ro/revista/12_7/surse/sumdiv.cpp

P050217: Sistem

Problema poate fi enunțată în termeni ai teoriei grafurilor astfel: *Să se determine numărul grafurilor neorientate distincte în care fiecare nod are gradul 2.*

Este ușor de determinat numărul grafurilor conexe cu proprietatea cerută; practic, orice permutare $(p_1, p_2, ..., p_N)$ a mulțimii poate duce la obținerea unei soluții. Dacă unim printr-o muchie vârfurile p_i și p_{i+1} ($i = 1, ..., n - 1$) și vârfurile p_1 și p_N obținem un graf (*hamiltonian*) în care toate vârfurile au gradul 2. Luând în considerare toate cele $N!$ permutări obținem $N!$ grafuri care respectă proprietatea cerută. Totuși, nu toate aceste grafuri sunt distincte; dacă alegem o anumită permutare atunci toate celelalte permutări circulare ale acesteia duc la obținerea aceluiași graf. De exemplu, permutările $(1, 2, ..., N)$ și $(N, 1, ..., N - 1)$ corespund aceluiași graf. Există N permutări circulare ale unei permutări date, deci numărul total al grafurilor distincte se reduce de N ori, devenind $(N - 1)!$. În plus, dacă parcurgem în ordine inversă o anumită permutare obținem același graf deoarece grafurile sunt neorientate. De exemplu, permutările $(1, 2, ..., N)$ și $(1, N, ..., 2)$ duc la obținerea aceluiași graf. Fiecare permutare poate fi parcursă în sens invers într-un singur mod, deci numărul total al grafurilor distincte se reduce de 2 ori. În concluzie, numărul grafurilor conexe cu proprietatea cerută este $(N - 1)! / 2$. Această valoare este valabilă pentru grafuri care conțin cel puțin trei noduri; nu există grafuri cu proprietatea cerută care să conțină unul sau două noduri.

Urmează să determinăm numărul total al grafurilor (conexe și neconexe) care au proprietatea cerută. Vom nota cu Nr_k numărul grafurilor care respectă proprietatea cerută și conțin k noduri. Vom avea $Nr_1 = Nr_2 = 0$ deoarece nu există nici un astfel de graf care să conțină unul sau două noduri. Prin convenție vom considera $Nr_0 = 1$ (se poate considera că graful vid respectă proprietatea cerută deoarece am putea spune că "*toate cele 0 noduri au gradul 2*", deși această afirmație este, practic, lipsită de sens; totuși această convenție se va dovedi utilă pentru simplificarea calculelor).

Vom considera, pe rând, că nodul 1 face parte din componente conexe cu 3, 4, ..., N noduri. Vom determina numărul de grafuri în care acest nod face parte din componente conexe cu k elemente (k este cuprins între 3 și N) și apoi vom aduna aceste valori.

Dacă nodul 1 face parte dintr-o componentă conexă cu k elemente, atunci există C_{N-1}^{k-1} posibilități de a alege celelalte k - 1 noduri care fac parte din aceeași componentă conexă. Folosind rezultatul demonstrat anterior, deducem că numărul componentelor conexe cu k elemente care respectă proprietatea cerută este $(k - 1)! / 2$. Indiferent care sunt cele k noduri care formează componenta conexă, celelalte $N - k$ noduri pot forma Nr_{N-k} grafuri cu proprietatea cerută. În concluzie, dacă nodul 1 face parte dintr-o componentă conexă cu k elemente, avem $Nr_{N-k} \cdot C_{N-1}^{k-1} \cdot \frac{(k-1)!}{2}$ grafuri distincte cu proprietatea cerută.

Se observă utilitatea convenției $Nr_0 = 1$ deoarece, pentru componenta conexă formată din toate cele N noduri ale grafului, obținem:

$Nr_{N-N} \cdot C_{N-1}^{N-1} \cdot \frac{(N-1)!}{2} = 1 \cdot 1 \cdot \frac{(N-1)!}{2} = \frac{(N-1)!}{2}$, adică exact numărul grafurilor conexe cu proprietatea cerută. Cazul $k = N$ ar fi putut fi tratat ca fiind unul particular, dar folosirea convenției duce la simplificarea formulelor de calcul.

Numărul total al grafurilor este obținut prin însumarea acestor valori pentru $k = 3, ..., N$, deci vom avea $Nr_N = \sum_{k=3}^{N} Nr_{N-k} \cdot C_{N-1}^{k-1} \cdot \frac{(k-1)!}{2}$ grafuri cu proprietatea cerută.

Chiar și pentru valori mici ale lui N, acest număr nu va putea fi reprezentat folosind tipuri standard de date, deci vor trebui simulate operații aritmetice cu numere mari. Folosind un artificiu de calcul, nu va trebui să implementăm decât adunarea a două numere mari și înmulțirea dintre un număr mare și un număr mai mic decât N.

Vom avea nevoie de adunarea numerelor mari pentru calcularea sumei care duce la obținerea valorilor Nr_k. Valorile însumate sunt obținute prin înmulțirea unui număr mare (Nr_{N-k} cu mai multe numere mai mici decât 100). Aceste numere sunt obținute folosind următorul artificiu:

$$C_{N-1}^{k-1} \cdot \frac{(k-1)!}{2} = \frac{(N-1)!}{(k-1)!(N-k)!} \cdot \frac{(k-1)!}{2} = \frac{(N-1)!}{2 \cdot (N-k)!} =$$
$$= \frac{(N-k+1) \cdot (N-k+2) \cdot K \cdot (N-1)}{2}$$

Așadar, șirul numerelor cu care se vor realiza înmulțiri este $N - k + 1, N - k + 2, N - 1$. Deoarece k este cel puțin 3, acest șir are cel puțin două elemente; elementele șirului sunt numere consecutive, deci unul dintre primele două este par. Acest număr par va fi împărțit cu 2 și se va obține un șir de elemente care vor fi înmulțite, pe rând, cu un număr mare.

Pentru determinarea valorii Nr_N este necesară calcularea tuturor valorilor Nr_k ($k < N$): acestea vor fi păstrate

într-un șir de numere mari, iar în final valoarea Nr_N va fi scrisă în fișierul de ieșire.

Analiza complexității

Fiecare operație aritmetică se efectuează în timp constant deoarece numărul cifrelor unui număr (dacă se folosește baza 10000) este mai mic decât 100.

Citirea datelor de intrare este realizată în timp *constant* pentru că se citește un singur număr.

Pentru calcularea unei valori Nr_K avem nevoie de însumarea a k - 3 termeni. Valorile Nr_i necesare sunt cunoscute de la pașii anteriori; pentru determinarea termenilor care trebuie însumați avem nevoie de i - 1 înmulțiri. Așadar, în total vom avea $(k - 4) + (k - 5) + ... + 1 = (k - 4)\cdot(k - 3)/2$ înmulțiri. În concluzie, ordinul de complexitate al operației de determinare a numărului Nr_k este $O(k^2)$. Vom determina N astfel de numere, ordinul de complexitate al întregii operații fiind $O(N^3)$ deoarece avem $k = O(N)$.

Datele de ieșire constau într-un singur număr mare, deci operația de scriere a acestora se realizează în timp *constant*.

În concluzie, algoritmul de rezolvare a acestei probleme are ordinul de complexitate $O(1) + O(N^3) + O(1) = O(N^3)$.

Dificultate **34**

Algoritmi	★☆☆☆☆
Structuri de date	★☆☆☆☆
Originalitate	★★☆☆☆
Cunoștințe	★★★★☆
Implementare	★★☆☆☆

www.ginfo.ro/revista/12_7/surse/sistem.cpp

P050218: Comitat

O posibilitate de rezolvare a acestei probleme este folosirea metodei programării dinamice. Vom încerca să determinăm un tablou tridimensional A, unde A_{ijk} va indica lungimea minimă a unui traseu care pornește din *Mordor* (turnul 0), ultimele două turnuri de pe acest traseu sunt j și k și traseul conține în total i turnuri.

Se observă că lungimile traseelor care conțin i turnuri pot fi determinate cu ajutorul lungimilor traseelor care conține i - 1 turnuri. Astfel, vom lua în considerare toate traseele pe care se află i - 1 turnuri, și ultimul turn al traseului este j și îl vom alege pe cel mai scurt dintre acestea. La lungimile acestor trasee vom adăuga distanța dintre turnurile j și k. Datorită faptului că un călăreț se poate deplasa doar paralel cu axele de coordonate, distanța dintre două turnuri i și j va fi dată de $|x_i - x_j| + |y_i - y_j|$. Fie p, penultimul turn de pe traseul care se termină în turnul j; evident, unghiul determinat de turnurile p, i și j nu trebuie să anuleze convexitatea traseului. Pentru a putea reconstitui traseul vom păstra, pentru fiecare triplet (i, j, k) numărul de ordine p al turnului care se află pe traseu înaintea turnurilor j și k.

În final, vom obține toate elementele tabloului tridimensional A. Evident, traseul trebuie să fie un poligon închis, deci ultimul turn trebuie să fie 0 (*Mordor*). Așadar, vom studia valorile A_{ij0}; vom determina cea mai mare valoare i pentru care există o valoare j astfel încât valoarea A_{ij0} este cel mult egală cu lungimea maximă a traseului.

După identificarea valorilor i și j vom putea determina turnurile de pe traseu cu ajutorul predecesorilor păstrați pentru fiecare triplet (i, j, k).

Pentru a mări viteza de execuție a programului vom determina, pentru fiecare triplet (i, j, k) dacă cele trei turnuri corespunzătoare se pot afla pe poziții consecutive în traseu (se respectă condiția de convexitate) și vom folosi aceste informații pe parcurs în loc să verificăm convexitatea de fiecare dată.

Inițial putem determina toate traseele pe care se află două turnuri (*Mordor* și oricare dintre celelalte). În continuare vom aplica algoritmul pentru a determina lungimile traseului pe care se află 3 turnuri, 4 turnuri etc.

Analiza complexității

Datele de intrare constau în numărul turnurilor, coordonatele acestora și lungimea maximă a traseului, deci ordinul de complexitate al operației de citire este $O(n)$.

Va trebui să verificăm pentru fiecare triplet (i, j, k) dacă se respectă condiția de convexitate, operație care are ordinul de complexitate $O(n^3)$, deoarece există $(n + 1)^3$ astfel de triplete (i, j și k variază între 0 și n).

Pentru a determina o valoarea A_{ijk} vor trebui luați în considerare toți predecesorii posibili, deci ordinul de complexitate al acestei operații este $O(n)$. Deoarece tabloul este tridimensional sunt realizate $O(n^3)$ astfel de operații, deci ordinul de complexitate al operației de determinare a tabloului A (și a tabloului care conține predecesorii) este $O(n)\cdot O(n^3) = O(n^4)$.

Identificarea valorii A_{ij0} care indică lungimea traseului care conține cele mai multe turnuri se realizează în timp ***pătratic*** deoarece, în cel mai defavorabil caz, sunt luate în considerare toate perechile (i, j).

Refacerea traseului (și scrierea turnurilor de pe traseu în fișierul de ieșire) pe baza tabloului predecesorilor se realizează în timp ***liniar***, deoarece vor exista cel mult n turnuri pe acest traseu.

În concluzie, algoritmul de rezolvare a acestei probleme are ordinul de complexitate $O(n) + O(n^3) + O(n^4) + O(n^2) + O(n) = O(n^4)$.

Dificultate **43**

Algoritmi	★★★☆☆
Structuri de date	★☆☆☆☆
Originalitate	★★★☆☆
Cunoștințe	★★☆☆☆
Implementare	★★☆☆☆

www.ginfo.ro/revista/12_7/surse/comitat.cpp

Košice, Slovacia

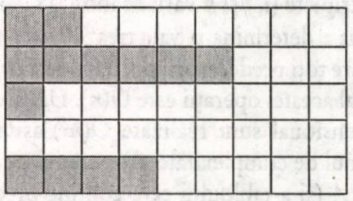
CEOI 2002

În continuare vă vom prezenta soluțiile oficiale ale celor șase probleme propuse spre rezolvare la ediția din acest an a Olimpiadei de Informatică a Europei Centrale.

P060207: Bugs Integrated, Inc.

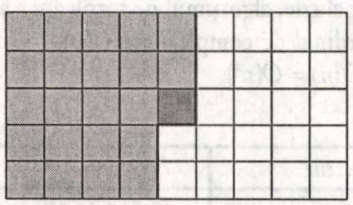
În prezentarea acestei soluții vom considera că valorile coordonatelor orizontale variază între 0 și N - 1, iar cele ale coordonatelor verticale între 0 și M - 1.

Fie $B = (b_0, b_1, ..., b_{M-1})$ un vector de numere întregi (numit *margine*). Definim mulțimea mărginită $S(B)$ ca fiind formată din toate celulele de coordonate $[x, y]$ astfel încât $x \le b_y$. Cu alte cuvinte, mulțimea marginii este formată din toate celulele aflate la stânga marginii. De obicei, nu se va face distincție între o margine și mulțimea sa.



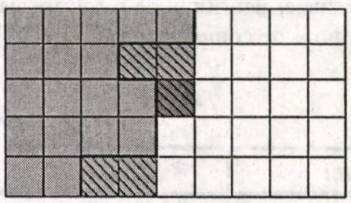
Mulțimea marginii B = (1, 5, 0, -7, 3) pentru N = 9 și M = 5

Pentru fiecare pereche $[i, j]$ ($0 \le i < N$, $0 \le j < M$) definim $[i, j]$-*marginea* ca fiind $B[i, j] = (b_0, b_1, ..., b_{M-1})$, astfel încât $b_0 = b_1 = ... = b_j = i$ și $b_{j+1} = b_{j+2} = ... = b_{M-1} = i - 1$. Mulțimea $S(B[i, j])$ va fi numită $[i, j]$-*bază*. Cu alte cuvinte, $S(B[i, j])$ este mulțimea celulelor care se află la stânga celulei de coordonate $[i, j]$ sau pe aceeași coloană, dar deasupra celulei $[i, j]$. Celula $[i, j]$ este inclusă și ea în $[i, j]$-*bază*.



[4, 2]-marginea B[4, 2] = (4, 4, 4, 3, 3) și baza sa

Considerăm o bază cu marginea $B[i, j]$. Fie P un vector $(p_0, p_1, ..., p_{M-1})$ ale cărui elemente sunt cuprinse în mulțimea {0, 1, 2}. Un astfel de vector este numit *profil*. Vom nota prin $B - P$ vectorul $(b_0 - p_0, ..., b_{M-1} - p_{M-1})$. Pentru *baza* considerată vom defini *profilul* P ca fiind mulțimea $S(B[i, j] - P)$. Vom nota cu 0 *profilul* pentru care toate valorile p_j sunt nule și cu e_j *profilul* pentru care avem $p_j = 1$, iar restul valorilor sunt nule.



Profilul P=(0, 2, 1, 0, 2) al [4, 2]-bazei

Un *profil* poate fi privit ca un număr cuprins între 0 și 3^M - 1, scris în baza 3.

Putem privi placa de silicon ca fiind mulțimea pătrățelelor valide (care nu sunt defecte). Vom nota această mulțime prin G. Așadar, va trebui să determinăm numărul maxim de *chip*-uri care pot fi tăiate dintr-o placă G.

Ideea care stă la baza rezolvării este folosirea programării dinamice; modalitatea în care este aplicată această metodă va fi descrisă în continuare.

Pentru fiecare *bază* $B[i, j]$ și fiecare *profil* P calculăm valoarea $A[i, j, P]$ care reprezintă numărul maxim de *chip*-uri care pot fi tăiate din mulțimea $G \cap S(B[i, j] - P)$. Se observă că avem $G \cap S(B[N - 1, M - 1] - 0) = G$, așadar soluția problemei va fi dată de valoarea $A[N - 1, M - 1, 0]$.

Secțiunea din placă ce corespunde unei mulțimi $G \cap S(B[0, j])$ este prea subțire pentru a se putea tăia vreun *chip* din ea, motiv pentru care vom avea $A[0, j, P] = 0$ pentru orice valori j și P.

În continuare vom lua în considerare toate bazele $B[i, j]$ parcurgând placa de la stânga la dreapta și de sus în jos. Pentru fiecare bază vom lua în considerare toate *profilurile* posibile.

Considerăm o *bază* $B[i, j]$ și un *profil* P; vom lua în considerare două situații: $p_j > 0$ și $p_j = 0$.

Dacă $p_j > 0$, atunci $G \cap S(B[i, j] - P) = G \cap S(B' - P')$, unde $B' = S(B[i, j])$ este *baza* luată în considerare la pasul

anterior, și $P' = P - e_j$. Datorită egalității anterioare, obținem $A[i, j, P] = A[i', j', P']$.

Dacă $p_j = 0$, există trei posibilități de a obține numărul maxim de *chip*-uri care pot fi tăiate din porțiunea $G \cap S(B - P)$ a plăcii:

- nu tăiem nici un *chip* care să aibă colțul din dreapta-jos în poziția $[i, j]$;
- tăiem un *chip* orizontal având colțul din dreapta-jos în pozițiile $[i, j]$;
- tăiem un *chip* vertical care să aibă colțul din dreapta-jos în poziția $[i, j]$.

Pentru primul caz, numărul maxim de *chip*-uri este dat de $A[i', j', P]$, unde $S(B[i', j'])$ este *baza* luată în considerare la pasul anterior.

Pentru al doilea caz, numărul de *chip*-uri este dat de $A[i'', j'', P + 2 \cdot e_j + 2 \cdot e_{j-1}]$, unde $S(B[i'', j''])$ este *baza* luată în considerare cu doi pași înainte.

Pentru al treilea caz, numărul maxim de *chip*-uri este dat de $A[i''', j''', P + e_j + e_{j-1} + e_{j-2}]$, unde $S(B[i''', j'''])$ este *baza* luată în considerare cu trei pași înainte.

Este evident că $A[i, j, P]$ va fi maximul acestor trei valori (al doilea și al treilea caz vor fi luate în considerare numai dacă *chip*-ul corespunzător poate fi tăiat în această poziție - nici unul dintre cele șase pătrățele pe care ar trebui să le ocupe *chip*-ul nu poate fi defect).

Testarea posibilităților de a tăia un *chip* vertical sau orizontal în diferite cazuri este o operație destul de simplă. Un *chip* orizontal poate fi tăiat dacă pătrățelele pe care le ocupă nu sunt defecte, avem $i \geq 1$, $j \geq 2$ și $p_j = p_{j-1} = 0$. În mod similar, un *chip* vertical va putea fi tăiat dacă pătrățelele pe care le ocupă nu sunt defecte, avem $i \geq 1$, $j \geq 2$ și $p_j = p_{j-1} = p_{j-2} = 0$. În final vom afișa valoarea $A[M - 1, N - 1, 0]$.

Se observă că nu trebuie reținute informații decât pentru ultimele trei *baze* considerate anterior și pentru *baza* curentă.

Analiza complexității

Operația de citire a datelor referitoare la o placă are ordinul de complexitate $O(M \cdot N)$ deoarece pot exista cel mult $M \cdot N$ pătrățele defecte.

Pentru fiecare element $A[i, j, P]$, timpul necesar calculării valorii sale este ***constant*** deoarece implică doar testarea validității unui număr finit de pătrățele și a unor condiții simple referitoare la i, j, p_j, p_{j-1} și p_{j-2}.

Fiecare *profil* este un vector format din M elemente care pot avea, fiecare trei valori distincte. Ca urmare, vom avea 3^M *profiluri* distincte, deci 3^M valori posibile pentru P. Vor fi $M \cdot N$ valori posibile pentru combinațiile i și j, deci tabloul tridimensional A va avea $M \cdot N \cdot 3^M$ elemente. Fiindcă fiecare element este determinat în timp constant, ordinul de complexitate al operației de determinare a numărului maxim de *chip*-uri care trebuie tăiate este $O(M \cdot N \cdot 3^M)$.

Datele de ieșire constau într-un singur număr, deci ordinul de complexitate al operației de scriere a acestora este $O(1)$.

În concluzie, pentru o anumită placă, ordinul de complexitate al algoritmului de rezolvare este $O(M \cdot N) + O(M \cdot N \cdot 3^M) + O(1) = O(M \cdot N \cdot 3^M)$.

Deoarece sunt D plăci, algoritmul va fi aplicat de D ori, dar valorile M și N nu vor fi aceleași, deci nu putem exprima simplu ordinul total de complexitate.

	Dificultate 52
Algoritmi	★★★☆☆
Structuri de date	★☆☆☆☆
Originalitate	★★☆☆☆
Cunoștințe	★★★★★
Implementare	★☆☆☆☆

http://cs.science.upjs.sk/ceoi/documents/bugs_sou.zip

P060208: Batalionul Cuceritorului

Pentru început vom încerca să estimăm cât de bună este o poziție. Dacă avem doar doi soldați și aceștia se află pe aceeași treaptă, atunci la runda următoare vom avea cel mult un soldat, dar care se află cu o treaptă mai sus. Așadar, putem spune că un soldat aflat pe treapta S este echivalent (are aceeași valoare) cu doi soldați aflați pe treapta $S + 1$.

Vom considera că un soldat aflat pe treapta S are valoarea 2^{N-S}. Valoarea unei poziții va fi dată de suma valorilor tuturor soldaților. Se observă că toate pozițiile în care cuceritorul a câștigat au cel puțin valoarea 2^{N-1} deoarece există un soldat pe prima treaptă.

Vom demonstra acum că în cazul în care valoarea unei poziții este mai mică decât 2^{N-1} și comandantul joacă optim, atunci cuceritorul va pierde. Dacă valoarea este mai mică decât 2^{N-1}, cuceritorul nu a câștigat încă. La o anumită rundă, el își va împărți soldații într-un mod oarecare. Valoarea unuia dintre cele două grupuri va fi obligatoriu mai mică decât 2^{N-2}. Comandantul va alege ca acest grup să rămână, iar celălalt să plece. Când un soldat urcă o treaptă, valoarea sa se dublează; ca urmare, valoarea noii poziții va fi mai mică decât $2 \cdot 2^{N-2} = 2^{N-1}$ și numărul soldaților va scădea. Există un număr finit de soldați, deci jocul se va termina după un număr finit de runde. Valoarea pozițiilor va fi întotdeauna mai mică decât 2^{N-1}, deci cuceritorul nu va putea câștiga; până la urmă, datorită faptului că numărul soldaților scade în continuu, nu va mai rămâne nici un soldat, deci cuceritorul va pierde.

Acum vom arăta că în cazul în care valoarea unei poziții este cel puțin egală cu 2^{N-1} și cuceritorul joacă optim, atunci el va câștiga. Cuceritorul va trebui să își împartă soldații în două grupuri a căror valoare este cel puțin egală cu 2^{N-2}. Indiferent de alegerea comandantului, noua poziție (după ce soldații rămași urcă o treaptă) va avea valoarea cel puțin egală cu 2^{N-1} și numărul soldaților va scădea. Numărul soldaților este finit, deci jocul se va încheia după un număr finit de mutări. Datorită faptului că valoarea pozițiilor nu va scădea niciodată sub 2^{N-1}, înseamnă că și valoarea ultimei poziții va fi cel puțin 2^{N-1}. Aceasta înseam-

nă că cel puțin un soldat se află pe prima treaptă, deci cuceritorul a câștigat.

Mai trebuie să arătăm că există întotdeauna posibilitatea de a împărții soldații în două grupuri a căror valoare să fie cel puțin egală cu 2^{N-2}. Considerăm șirul $a_1, ..., a_M$ astfel încât $N - 2 \geq a_1 \geq a_2 \geq ... \geq a_M \geq 0$, $M \geq 2$ și $\sum_{i=1}^{M} 2^{a_i} \geq 2^{N-2}$; vom arăta că există o valoare $k \leq M$ astfel încât $\sum_{i=1}^{k} 2^{a_i} \geq 2^{N-2}$. Pentru a demonstra această afirmație vom folosi metoda inducției matematice. Pentru $M = 2$ avem $a_1 = N - 2$ și orice valoare a_2 cuprinsă între 0 și $N - 2$ sau $a_1 = a_2 = N - 3$. În primul caz alegem $k = 1$, iar în al doilea $k = 2$ pentru a obține $\sum_{i=1}^{k} 2^{a_i} \geq 2^{N-2}$. Așadar, ipoteza de inducție este demonstrată. Pentru $M > 2$, dacă $\sum_{i=1}^{M} 2^{a_i} = 2^{N-2}$ alegem $k = M$. Dacă avem $M > 2$ și $\sum_{i=1}^{M} 2^{a_i} > 2^{N-2}$, atunci, datorită faptului că 2^{N-2} și $\sum_{i=1}^{M} 2^{a_i}$ sunt multiplii ai lui 2^{a_M}, avem $\sum_{i=1}^{M} 2^{a_i} \geq 2^{N-2} + 2^{a_M}$, deci $\sum_{i=1}^{M-1} 2^{a_i} \geq 2^{N-2}$. În caz de egalitate ne oprim, iar în caz de inegalitate aplicăm aceeași metodă pentru $M - 1$. Vom ajunge fie în situația $M = 2$ (corespunzătoare ipotezei de inducție), fie ne vom opri deoarece am obținut egalitatea. Așadar afirmația este demonstrată.

Rezultă că, dacă valoarea totală a soldaților este cel puțin egală cu 2^{N-1}, vom putea alege primii soldați (cei de pe cele mai înalte trepte) a căror sumă a valorilor este exact 2^{N-2}. Cuceritorul va putea împărți soldații în două grupuri cu valori cel puțin egale cu 2^{N-2} (primul format din acești soldați, iar celălalt din cei rămași), deci până la urmă va câștiga jocul.

Pentru că pot fi până la 2000 de trepte, vom fi nevoiți să simulăm operații aritmetice cu numere mari sau să folosim diferite artificii pentru a le evita (valoarea unui soldat va putea atinge 2^{1999}, iar cea a unei poziții $1000000000 \cdot 2^{1999} \approx 2^{30} \cdot 2^{1999} = 2^{2029}$).

Dacă suntem într-o poziție câștigătoare, vom alege soldați de pe treptele 1, 2, ... până în momentul în care valoarea lor totală ajunge la 2^{N-2}. Dacă suntem într-o situație necâștigătoare vom alege un grup vid de soldați și vom pierde imediat.

Pentru a mări viteza algoritmului putem precalcula, la începutul jocului, anumite valori. Fie s_i numărul soldaților de pe cea dea i-a treaptă. Vom precalcula sumele $T_k = \sum_{i=1}^{k} s_i \cdot 2^{N-i}$. Numerotăm soldații de 1 la M în ordinea în care se află pe trepte la începutul jocului. Dacă folosim algoritmul prezentat, vom putea împărți soldații astfel încât fiecare grup să fie format din soldați cu numere de ordine consecutive. Vom păstra numărul curent de soldați de pe fiecare treaptă, precum și două referințe la primul și ultimul soldat din grupul curent. Se observă că nu va trebui să mutăm soldații de pe o treaptă pe alta, deoarece știm că după r runde, un soldat se va afla cu r trepte mai sus față de poziția lui inițială.

Folosind valorile precalculate, vom găsi foarte repede valoarea unui grup de soldați cu numere de ordine consecutive. Pentru a determina poziția în care va fi împărțit grupul se poate folosi căutarea binară.

Analiza complexității

Datorită faptului că numărul de cifre al numerelor mari este limitat, putem considera că ordinul de complexitate al operațiilor cu astfel de numere este $O(1)$.

Preluarea datelor de intrare constă în completarea de către bibliotecă a unui tablou cu N elemente, deci se realizează în timp *liniar*.

Numărul de runde este cel mult $N - 1$, deoarece soldații de pe ultima treaptă vor ajunge pe prima după $N - 1$ runde. La fiecare rundă vom realiza o căutare binară care are ordinul de complexitate $O(\log N)$ și vom completa un tablou cu N elemente, operație realizabilă în timp *liniar*. Apelul funcției `step` implică efectuarea unor operații în timp *liniar*, funcția apelându-se o dată pentru fiecare rundă. Așadar, operațiile realizate pentru fiecare rundă au ordinul de complexitate $O(\log N) + O(N) + O(N) = O(N)$. Pentru toate rundele ordinul de complexitate devine $O(N) \cdot O(N) = O(N^2)$.

Precalcularea valorilor T_k se realizează în timp *liniar* deoarece fiecare valoare T_k se obține folosind valoarea T_{k-1} și efectuând o adunare și o înmulțire: $T_k = T_{k-1} + s_k \cdot 2^{N-k}$.

Datele de ieșire sunt create de funcția `step`, deci scrierea lor nu implică operații suplimentare.

Așadar, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(N) + O(N^2) + O(N) = O(N^2)$.

Dificultate 42

Algoritmi	★★☆☆☆
Structuri de date	★☆☆☆☆
Originalitate	★★★☆☆
Cunoștințe	★★★★☆
Implementare	★★☆☆☆

http://cs.science.upjs.sk/ceoi/documents/conquer_sou.zip

P060209: Un gărduleț decorativ

Pentru rezolvarea acestei probleme vom aplica metoda programării dinamice.

Fie $U_{N,i}$ numărul permutărilor care descriu un gard în care primul element al permutării este i, iar al doilea este mai mare decât primul. O astfel de permutare va fi numită *crescătoare*. Analog, fie $D_{N,i}$ numărul permutărilor care descriu un gard în care primul element al permutării este i, iar al doilea este mai mic decât primul. O astfel de permutare este numită *descrescătoare*.

Deoarece N este cuprins între 1 și 20, putem precalcula valorile U și D pentru toate combinațiile de valori i și N.

Formulele de calcul sunt următoarele:

- $D_{N,1} = 0$
- $D_{N,i+1} = \sum_{k=1}^{i} U_{N-1,k} = D_{N,i} + U_{N-1,i}$
- $U_{N,i} = D_{N,N+1-i}$

Prima formulă este evidentă pentru că primul element este 1, iar al doilea nu poate fi mai mic decât acesta.

Cea de-a doua formulă se obține astfel: se consideră toate permutările descrescătoare de lungime N care încep cu $i + 1$. Al doilea element k al unei astfel de permutări va fi cuprins între 1 și i. Dacă eliminăm din permutare primul element ($i + 1$) și descreștem cu 1 toate elementele mai mari decât i, obținem o permutare crescătoare care începe cu k. Așadar există o corespondență biunivocă între mulțimea permutărilor crescătoare de lungime N care încep cu i și mulțimea permutărilor crescătoare care încep cu o valoare mai mică decât i. Ca urmare, cele două mulțimi au același număr de elemente.

Așadar avem:

$$D_{N,i+1} = \sum_{k=1}^{i} U_{N-1,k} = \sum_{k=1}^{i-1} U_{N-1,i} + U_{N-1,i} = D_{N,i} + U_{N-1,i}.$$

Cea de-a treia relație este obținută foarte ușor dacă punem în corespondență o permutare $(p_1, \ldots, p_N)$ cu permutarea $(N + 1 - p_1, \ldots, N + 1 - p_N)$. Există o corespondență biunivocă între mulțimea permutărilor crescătoare care încep cu i și cea a permutărilor descrescătoare care încep cu $N + 1 - i$. Ca urmare, cele două mulțimi au același număr de elemente.

Vom folosi valorile U și D pentru a determina gărdulețul cu numărul de catalog C.

Este evident că există $U_{N,i} + D_{N,i}$ permutări care încep cu i. Primul element al permutării se obține foarte simplu, fiind cel mai mic număr pentru care $\sum_{i=1}^{a_1} (D_{N,i} + U_{N,i}) \geq C$.

După determinarea acestui element vom calcula valoarea a_2. Vom scădea cu 1 valorile tuturor elementelor din permutarea pe care o căutăm care sunt mai mari decât a_1. Restul permutării va descrie un gărdulet cu $N - 1$ elemente. Știm că dacă primul element al noii permutări este mai mic decât a_1, va fi descrescătoare, iar dacă este mai mare decât a_1, va fi crescătoare. De asemenea, știm că trebuie să căutăm cea de-a C_1-a astfel de permutare unde:

$$C_1 = C - \sum_{i=1}^{a_1-1} (D_{N,i} + U_{N,i}).$$

Pentru x cuprins între 1 și $a_1 - 1$ există $U_{N-1,x}$ astfel de permutări care încep cu x, iar pentru $x > a_1$ există $D_{N-1,x}$ permutări care încep cu x. Vom determina următorul element y al permutării într-un mod similar determinării primului element. Dacă obținem $y < a_1$ vom alege $a_2 = y$, iar dacă avem $y > a_1$, vom alege $a_2 = y + 1$.

Acum cunoaștem primele două elemente ale permutării pe care o căutăm și știm dacă aceasta este crescătoare sau descrescătoare. De asemenea cunoaștem numărul de ordine C_2 al permutării de pe ultimele $N - 2$ poziții. Următoarele elemente ale permutării date vor fi determinate folosind aceeași metodă.

Presupunem că știm care sunt primele k ($k \geq 2$) elemente ale permutării pe care o căutăm și numărul de ordine C_k al permutării de pe ultimele $N - k$ poziții. Ultimele două elemente cunoscute arată dacă permutarea de lungime $N - k$ este crescătoare sau descrescătoare. Vom determina primul element al acestei permutări luând în considerare toate permutările valide de $N - k$ elemente care încep cu x până depășim valoare C_k. Calculăm valoarea C_{k+1} prin scăderea din C_k a numărului permutărilor care încep cu o valoare mai mică decât elementul a_k determinat.

Pentru simplitate, vom lucra la fiecare pas k cu o permutare de $N - k$ elemente cuprinse între 0 și $N - k - 1$. Un element i dintr-o astfel de permutare va corespunde în permutarea finală celui de-al i-lea element care nu a fost folosit anterior (pe pozițiile anterioare ale permutării căutate).

Analiza complexității

Datele de intrare constau într-un singur număr, deci ordinul de complexitate al operației de citire a acestora este $O(1)$.

Precalcularea valorilor U și D implică un timp *constant* pentru fiecare valoare $U_{N,i}$ și $D_{N,i}$. În total sunt N^2 valori $U_{N,i}$ și N^2 valori $D_{N,i}$, deci ordinul de complexitate al operației de determinare a acestora este $O(N^2) + O(N^2) = O(N^2)$.

Determinarea unui element al permutării căutate se realizează în timp *liniar* deoarece implică luarea în considerare a cel mult N valori pentru acesta. În total sunt determinate N elemente, deci determinarea permutării căutate are ordinul de complexitate $O(N) \cdot O(N) = O(N^2)$.

Operația de afișare a datelor de ieșire constă în scrierea celor N elemente ale permutării, deci are ordinul de complexitate $O(N)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme pentru un gărdulet este $O(1) + O(N^2) + O(N^2) + O(N) = O(N^2)$.

Deoarece sunt k gărdulețe, algoritmul va fi aplicat de k ori, dar pentru valori diferite ale lui N, deci ordinul total de complexitate nu poate fi exprimat printr-o expresie simplă.

Dificultate **35**

Algoritmi	★★☆☆☆
Structuri de date	★☆☆☆☆
Originalitate	★☆☆☆☆
Cunoștințe	★★★☆☆
Implementare	★☆☆☆☆

http://cs.science.upjs.sk/ceoi/documents/fence_sou.zip

P060210: Autostrada și cei șapte pitici

Pentru început vom observa faptul că o autostradă poate fi construită dacă și numai dacă nu intersectează înfășurătoarea convexă a punctelor care reprezintă casele piticilor. Dacă este intersectată înfășurătoarea convexă, atunci ea va fi

intersectată în două dintre muchii, deoarece autostrăzile nu pot trece prin punctele care reprezintă casele. Dacă autostrada intersectează o muchie (latură) a înfășurătorii convexe, atunci cele două case din extremitățile laturii se vor afla de o parte și de alta a autostrăzii, deci aceasta nu poate fi construită.

După citirea coordonatelor claselor, se construiește înfășurătoarea convexă a acestora, folosind unul dintre algoritmii clasici de realizare a acestei operații.

În continuare vom citi datele referitoare la o linie și va trebui să decidem dacă intersectează un poligon convex. Pentru a găsi un algoritm eficient care să realizeze această operație vom presupune că linia are o direcție și, potrivit acestei direcții, există o cea mai din stânga și o cea mai din dreapta casă. Este evident că linia va intersecta poligonul dacă și numai dacă cele două case (puncte) se află în semiplane diferite determinate pe linie.

În continuare vom prezenta un algoritm eficient care determină cel mai din stânga și cel mai din dreapta punct corespunzător unei direcții.

Să presupunem că știm care este cel mai din stânga și cel mai din dreapta punct pentru o anumită direcție. Dacă schimbăm direcția cu un unghi foarte mic, punctele vor rămâne aceleași.

Totuși, după mai multe schimbări de direcție, vor apărea modificări. Vom încerca să determinăm când are loc o astfel de modificare. Considerăm două linii care respectă direcția considerată și care trec prin cel mai din dreapta și cel mai din stânga punct. Când rotim direcția, cele două puncte nu se vor modifica decât în momentul în care una dintre laturile înfășurătorii convexe se va afla pe una dintre cele două linii. În acest moment celălalt punct care determină latura respectivă va deveni cel mai din dreapta sau cel mai din stânga punct. Așadar, cele două puncte căutate se modifică doar dacă direcția este aceeași cu cea a uneia dintre laturile înfășurătorii convexe.

Dacă înfășurătoarea convexă are M laturi, atunci mulțimea direcțiilor va fi împărțită în M intervale, fiecărui interval corespunzându-i o pereche de puncte (un cel mai din dreapta și un cel mai din stânga punct). Pentru fiecare dreaptă vom determina direcția folosind o căutare binară și vom verifica dacă punctele corespunzătoare intervalului direcției sunt sau nu în același semiplan determinat de dreaptă.

Analiza complexității

Datele de intrare constau în N puncte și M linii, deci ordinul de complexitate al operației de citire a acestora este $O(M) + O(N) = O(M + N)$.

Determinarea înfășurătorii convexe folosind un algoritm eficient are ordinul de complexitate $O(N \cdot \log N)$.

Înfășurătoarea convexă va conține cel mult N elemente, deci vom determina $O(N)$ intervale ale direcțiilor. Determinarea unui interval se realizează în timp *constant*, deci ordinul de complexitate al operației de determinare a tuturor intervalelor este $O(N)$.

Căutarea binară necesită un timp *logaritmic*, și, datorită faptului că realizăm M căutări, verificarea posibilităților de construire a tuturor autostrăzilor are ordinul de complexitate $O(M \cdot \log N)$.

Pentru fiecare autostradă vom scrie un singur mesaj, deci ordinul de complexitate al operației de scriere a datelor de ieșire este $O(M)$.

În concluzie, ordinul de complexitate al algoritmului de rezolvare a acestei probleme este $O(M + N) + O(N \cdot \log N) + O(N) + O(M \cdot \log N) + O(M) = O((M + N) \cdot \log N)$.

Dificultate 41	
Algoritmi	★★★☆☆
Structuri de date	★☆☆☆☆
Originalitate	★★☆☆☆
Cunoștințe	★★★★☆
Implementare	★★☆☆☆

http://cs.science.upjs.sk/ceoi/documents/dwarfs-sou.zip

P060211: Gărzile regale

Pentru simplitate, vom considera castelul ca fiind înconjurat de ziduri. Vom numi *segment de linie* o porțiune continuă a unei linii care nu conține nici un zid și nu poate fi extins. Adică este delimitată la stânga și la dreapta de ziduri.

Similar, vom numi *segment de coloană* o porțiune continuă a unei coloane care nu conține nici un zid și care nu poate fi extinsă. Este evident că fiecare segment de linie sau coloană poate conține cel mult un gardian.

Construim un graf bipartit în care vârfurile uneia dintre partiții corespund *segmentelor de linie* iar vârfurile celeilalte partiții corespund *segmentelor de coloană*. Între două noduri din partiții diferite va exista o muchie dacă și numai dacă segmentele respective se intersectează și punctul de intersecție nu conține o capcană.

Cu alte cuvinte, o muchie corespunde pătrățelelor în care se pot afla gardieni.

Considerăm acum o amplasare validă a gardienilor. Muchiile grafului bipartit care corespund unor poziții în care se află gardieni formează un cuplaj al grafului (fiecare *segment de linie* sau *coloană* conține cel mult un gardian, deci fiecare vârf poate fi incident cu cel mult o muchie aleasă). Pe de altă parte, fiecare cuplaj al grafului bipartit corespunde unei amplasări valide a gardienilor.

Așadar, pentru a amplasa cât mai mulți gardieni, este suficient să determinăm un cuplaj maxim în graful bipartit construit. Pentru determinarea cuplajului poate fi folosit unul dintre algoritmii clasici destinați acestui scop (vezi *CLR*[1] p. 515).

După determinarea cuplajului vom scrie în fișierul de ieșire numărul muchiilor care îl formează (adică numărul maxim al gardienilor), precum și pozițiile corespunzătoare acestor muchii.

1. *Cormen T. H., Leiserson C. E., Rivest R. L., Introducere în Algortmi, Computer Libris Agora, Cluj-Napoca, 2000*

Analiza complexității

Castelul este codificat printr-o matrice cu M linii și N coloane, deci operația de citire a datelor de intrare are ordinul de complexitate $O(M \cdot N)$.

Construirea grafului bipartit implică traversarea matricei, deci ordinul de complexitate al acestei operații este $O(M \cdot N)$.

Algoritmul de determinare a cuplajului maxim are ordinul de complexitate $O(|V| \cdot (|V| \cdot |E|))$, unde $|V|$ este numărul de vârfuri, iar $|E|$ este numărul de muchii. Graful construit are $O(M \cdot N)$ vârfuri și $O(M \cdot N)$ muchii, deci în acest caz ordinul de complexitate al algoritmului va fi $O(M^2 \cdot N^2)$.

Cuplajul va conține $O(M \cdot N)$ muchii, deci acesta va fi ordinul de complexitate al operației de scriere al datelor de ieșire.

În concluzie, algoritmul de rezolvare al acestei probleme are ordinul de complexitate $O(M \cdot N) + O(M \cdot N) + O(M^2 \cdot N^2) + O(M \cdot N) = O(M^2 \cdot N^2)$.

Dificultate 47

Algoritmi	★★★☆☆
Structuri de date	★★☆☆☆
Originalitate	★★★☆☆
Cunoștințe	★★★☆☆
Implementare	★☆☆☆☆

http://cs.science.upjs.sk/ceoi/documents/guards-sou.zip

P060212: Petrecere aniversară

Vom considera că numele prietenilor lui *John* sunt variabile logice. Dacă o variabilă este adevărată, înseamnă că *John* ar trebui să invite persoane corespunzătoare, iar dacă este falsă, persoana nu va fi invitată. De asemenea, cererile primite de *John* pot fi privite ca expresii logice. Sarcina noastră este de a atribui valori variabilelor astfel încât prin evaluarea tuturor formulelor să se obțină valoarea *adevărat*.

De fapt, aceasta este una dintre cele mai cunoscute probleme de algoritmică, ea fiind cunoscută sub numele de problema *SAT*, prescurtarea de la satisfiabilitate (vezi *CLR* p. 805). Pe cazul general, se știe că această problemă este *NP*-completă, deci nu se cunoaște nici un algoritm polinomial de rezolvare a acesteia.

Câteva dintre cele zece fișiere de intrare sunt destul de mari, deci fișierele de ieșire corecte nu pot fi determinate dacă se folosește un algoritm exponențial. Cu toate acestea, cele zece fișiere folosite pentru testare sunt cunoscute și au anumite particularități care permit determinarea fișierelor de ieșire corecte. Pot fi folosite orice mijloace pentru determinarea fișierelor de ieșire; totuși, pentru unele dintre ele, este necesară scrierea unor programe.

În cele ce urmează, literele A, B, ... vor fi folosite pentru a nota expresii logice arbitrare, nu doar variabile. Vom folosi semnul minus ('-') pentru a nota negația logică a unei expresii; așadar, vom putea nota anumite expresii logice care nu sunt permise de enunțul problemei (de exemplu $-(A|B)$). Se poate observa foarte ușor că semnul '&' reprezintă operatorul de conjuncție (*ȘI logic*), semnul '|' reprezintă operatorul de disjuncție (*SAU logic*), iar semnul '=>' reprezintă implicația logică.

Se observă că expresiile $(A \Rightarrow B)$, $(-A \mid B)$ și $(-B \Rightarrow -A)$ sunt echivalente. Ca urmare, expresia $(A \Rightarrow -A)$ este adevărată dacă și numai dacă expresia A este falsă. De asemenea, formula $(A \Rightarrow (B \Rightarrow C))$ este echivalentă cu $((A \,\&\, B) \Rightarrow C)$. Vom avea nevoie și de legile lui *Morgan*:

- $-(A_1 \,\&\, \ldots \,\&\, A_m)$ este echivalentă cu $(-A_1 \mid \ldots \mid -A_m)$;
- $-(A_1 \mid \ldots \mid A_m)$ este echivalentă cu $(-A_1 \,\&\, \ldots \,\&\, -A_m)$.

Din afirmațiile anterioare rezultă că următoarele expresii sunt echivalente:

- $(A_1 \Rightarrow (A_2 \Rightarrow (\ldots (A_m \Rightarrow (B_1 \mid \ldots \mid B_n)) \ldots)))$;
- $((A_1 \,\&\, \ldots \,\&\, A_m) \Rightarrow (B_1 \mid \ldots \mid B_n))$;
- $((-(A_1 \,\&\, \ldots \,\&\, A_m)) \mid (B_1 \mid \ldots \mid B_n))$;
- $(-A_1 \mid \ldots \mid -A_m \mid B_1 \mid \ldots \mid B_n)$.

Vom numi variabilele și negațiile acestora folosind termenul comun *literal*. Spunem că o expresie este în ***formă normală conjunctivă*** (*FNC*), dacă este o conjuncție între mai multe formule logice și fiecare dintre aceste formule logice este o disjuncție a unor literali. De exemplu, formula $((A \mid B) \,\&\, (B \mid -C \mid C \mid C) \,\&\, -D \,\&\, (A \mid C))$ este în *FNC*. Se poate demonstra matematic faptul că orice expresie logică este echivalentă cu o expresie în *FNC*. Aceste observații ne vor permite să rescriem anumite fișiere de intrare pentru ca expresiile corespunzătoare să fie în *FNC*.

Se observă că denumirile variabilelor din fișierele 2 - 10 sunt *b*, *c*, *d*, ..., *i*, *j*, *ba*, *bb* etc. Dacă înlocuim a *i*-a literă a alfabetului cu a *i*-a cifră (începând numerotarea cu cifra 0) obținem variabilele 1, 2, 3, ..., 8, 9, 10, 11 etc. În plus, negația este reprezentată de semnul minus, deci negația unei variabile va fi dată de numărul negativ obținut prin adăugarea semnului '-' în fața numărului care reprezintă variabila.

PARTY1.IN

Primul fișier nu respectă convenția de notare a variabilelor din celelalte 9 și cea mai simplă modalitate de obținere a fișierului de ieșire corect este rezolvarea manuală a acestui caz. Sunt doar șapte expresii și cinci variabile, deci nu va fi prea dificilă găsirea soluției.

PARTY2.IN - PARTY4.IN

Aproape toate expresiile din aceste trei fișiere de intrare au forma $(lit_1 \mid lit_2 \mid \ldots \mid lit_k)$, unde lit_x este un literal. Acest tip de fișier este destul de convenabil deoarece înseamnă că cel puțin un literal de pe fiecare linie trebuie să fie *adevărat*. Expresiile care nu respectă această regulă vor fi pur și simplu rescrise folosind formulele descrise anterior. Observăm că nu mai avem nevoie de caracterele '(', ')' sau '|', deci le putem elimina. Fișierul de intrare va fi privit ca fiind o conjuncție de linii și, după rescrierea liniilor care nu respectă regula, expresia va fi în *FNC*.

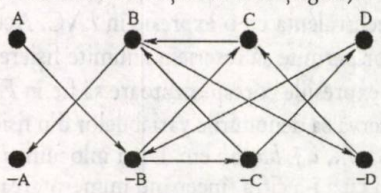
Aceste fișiere de intrare sunt relativ mici, deci pot fi utilizați algoritmi *backtracking* pentru a găsi soluțiile. Folosind câteva optimizări simple, soluțiile vor fi determinate foarte repede.

PARTY5.IN - PARTY9.IN

Aceste fișiere de intrare sunt prea mari pentru ca un algoritm exponențial să ruleze într-un timp rezonabil. Totuși, studiind cu atenție fișierele, observăm că fiecare expresie (cu excepția câtorva din fișierul `PARTY9.IN`) conține doar doi literari care formează o implicație sau o disjuncție.

Putem rescrie fiecare expresie astfel încât să devină o implicație. De exemplu (A | B) devine ($-A$ => B). În continuare construim un graf orientat ale cărui vârfuri vor fi literalii care apar în expresie (variabilele și negațiile lor), iar arcele vor reprezenta implicațiile. Semnificația unui arc va fi următoarea: dacă vârful din care pornește are valoarea *adevărat*, atunci și vârful la care ajunge trebuie să aibă valoarea *adevărat*.

Din expresia ($-A$ => B) deducem că există arc de la vârful corespunzător literalului -A la cel corespunzător literalului B. Expresia este echivalentă cu ($-B$ => A) deci vom avea un arc și de la $-B$ la A. Așadar, fiecare expresie implică apariția a două arce în graf. Se observă că graful este simetric (dacă înlocuim fiecare literal cu negația sa și inversăm sensul arcelor obținem același graf).



Graful pentru formulele: (A => -B), (B | -C), (B | C), (D => -B) și (C | D)

În continuare vom eticheta fiecare vârf cu *adevărat* sau *fals*, astfel încât pentru o variabilă A, exact unul dintre vârfurile A și $-A$ este etichetat cu *adevărat*. De asemenea, dacă un vârf V este etichetat cu *adevărat*, atunci toate vârfurile U pentru care există un drum de la V la U trebuie etichetate cu *adevărat*.

Evident, dacă pentru o anumită variabilă A, vârfurile A și $-A$ se află în aceeași componentă tare-conexă a grafului, nu există nici o etichetare care respectă cele două condiții amintite anterior (unul dintre acești doi literali trebuie să aibă valoarea *adevărat* și, în cazul în care fac parte din aceeași componentă tare-conexă, există un drum între nodurile celor doi literali, deci și celălalt trebuie să aibă valoarea *adevărat*; am ajuns astfel la o contradicție, deci afirmația inițială este falsă; așadar, A și $-A$ nu pot fi în aceeași componentă tare-conexă). În cele ce urmează vom arăta că există soluție în toate celelalte cazuri.

Vom alege o componentă tare-conexă pentru care nu există nici un arc care pornește dintr-un vârf din afara componentei și ajunge într-un vârf din interiorul ei. Vom eticheta toate vârfurile din această componentă C cu *fals*. Datorită simetriei grafului vârfurile care corespund negațiilor literalilor vor forma o componentă tare-conexă C' pentru care nu există nici un arc care pornește dintr-un vârf din interiorul ei și ajunge într-un vârf din afara ei. Vom eticheta toate vârfurile din C' cu *adevărat*. Este evident că etichetarea vârfurilor din C și C' este corectă și nu impune nici o restricție asupra etichetării celorlalte vârfuri ale grafului.

Așadar putem elimina din graf componentele C și C' și aplica același algoritm pentru restul grafului.

Numărul de noduri ale grafului este dublul numărului de variabile; numărul de arce este dublul numărului de expresii; așadar dimensiunea grafului este liniară în raport cu dimensiunea fișierului de intrare. Există un bine cunoscut algoritm liniar de determinare a componentelor tare-conexe ale unui graf. Vom construi apoi un graf al componentelor: fiecare componentă este reprezentată de un nod și există un arc între două noduri i și j, dacă în graful inițial există cel puțin un arc de la un nod din componenta i la un nod din componenta j. Se aplică o sortare topologică asupra grafului componentelor pentru a le alege pe cele care vor fi etichetate cu *fals* (și implicit pe cele ale căror vârfuri vor fi etichetate cu *adevărat*). Întregul algoritm se execută în timp liniar.

PARTY10.IN

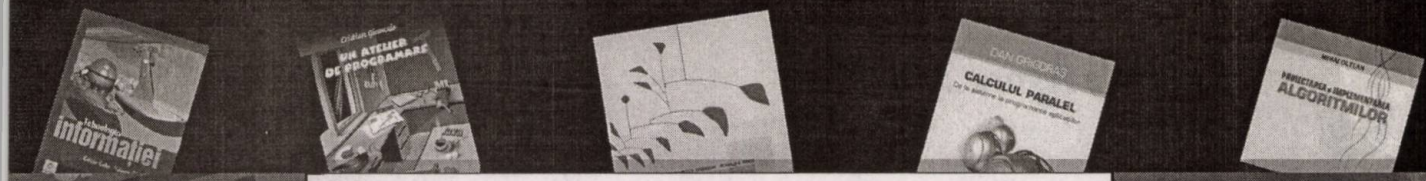
Acesta este cel mai mare și pare cel mai complicat dintre toate. La o privire mai atentă descoperim că, din contră, soluția poate fi obținută destul de repede. Dacă cercetăm fișierul, observăm că ultimele linii au forma (A => $-A$) sau ($-B$ => B). Din expresiile de primul tip deducem că valoarea variabilei A este *fals*, iar din cele de al doilea tip că valoarea variabilei B este *adevărat*.

Folosind această observație obținem valorile a 997 dintre cele 1000 de variabile. Cele trei valori rămase pot fi determinate folosind primele trei expresii din fișier.

Din enunțul problemei știm că există întotdeauna o soluție. Soluția determinată astfel este singura care verifică expresiile de pe liniile luate în considerare, așadar este singura soluție posibilă. Nu trebuie să verificăm și expresiile de pe celelalte linii deoarece știm că ele trebuie să fie *adevărate* (dacă, totuși, decidem să le verificăm, vom observa că, într-adevăr, toate acestea au valoarea *adevărat* dacă valorile variabilelor sunt determinate folosind primele trei expresii și ultimele 997).

Dificultate	65
Algoritmi	★★★☆☆
Structuri de date	★★☆☆☆
Originalitate	★★★☆☆
Cunoștințe	★★★★★
Implementare	★★★★☆

http://cs.science.upjs.sk/ceoi/documents/party-sou.zip



COMPUTER LIBRIS AGORA

Cartea este o culegere de studii de caz, construite pe temelia problemelor concursurilor Southeastern European Regional Programming Contest din perioada 1995-1999, concursuri organizate sub egida Association for Computing Machinery.
Cartea se adresează atât începătorilor, cât și celor cu experiență în domeniul programării, în încercarea de a contura prin exemple imaginea unei lumi diverse și pasionante, dar pretențioase, aceea a înzestrării mașinii de calcul cu capacitatea de a rezolva corect și eficient.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

Etapa FINALĂ

Vă prezentăm soluțiile oficiale ale celor cinci probleme propuse spre rezolvare la etapa finală a celei de-a treia ediții a concursului internațional de programare Bursele Agora, organizat de redacția GInfo și editura Agora Media.

P060219: Conducte

Putem privi sistemul de rafinării și benzinării ca fiind un graf neorientat complet ale cărui noduri reprezintă obiectivele (benzinării sau rafinării), iar costurile muchiilor sunt date de distanțele (lungimile conductelor) care leagă două obiective. Evident, nu putem uni printr-o conductă două rafinării, așadar vom considera că muchiile dintre rafinării au costul 0 (nu sunt necesare conducte pentru ca benzina să ajungă de la o rafinărie la alta deoarece există deja benzină în toate rafinăriile).

Acum, problema se reduce la a determina un arbore parțial de cost minim în graful construit. Pentru aceasta vom folosi *algoritmul lui Prim* (vezi *CLR*[1], p. 435); în final vom afișa muchiile care fac parte din arborele parțial minim dar nu au ca extremități două noduri care corespund unor rafinării.

Datorită faptului că putem avea până la 1000 de benzinării nu vom putea reprezenta în memorie graful folosind o matrice de adiacență. De asemenea, nu este posibilă păstrarea unei liste de muchii deoarece numărul acestora este prea mare. De aceea, vom păstra doar coordonatele benzinăriilor și rafinăriilor și vom calcula costurile muchiilor (distanțele dintre obiective) în momentul în care avem nevoie de aceste informații. Din acest motiv, nu poate fi folosit *algoritmul lui Kruskal* de determinare a arborelui parțial minim pentru că acesta implică sortarea tuturor muchiilor, deci păstrarea acestora în memorie.

Analiza complexității

Datele de intrare constau în coordonatele rafinăriilor și benzinăriilor, deci operația de citire a acestora are ordinul de complexitate $O(n) + O(k) = O(n + k)$.

Este obligatoriu ca nodurile corespunzătoare rafinăriilor să fie legate între ele în arborele parțial minim folosind muchii de cost 0, operație care se realizează în timp liniar (ordinul de complexitate este $O(k)$). Pentru celelalte noduri vom folosi *algoritmul lui Prim*, ordinul de complexitate al acestuia fiind $O(n^2)$. Așadar, determinarea arborelui parțial minim are ordinul de complexitate $O(n^2) + O(k) = O(n^2 + k)$.

Datele de ieșire constau în cele n muchii care leagă benzinăriile între ele sau de rafinării, deci ordinul de complexitate al operației de scriere a acestora este $O(n)$.

În concluzie, algoritmul de rezolvare a acestei probleme are ordinul de complexitate $O(n + k) + O(n^2 + k) + O(n) = O(n^2 + k)$.

Dificultate 29	
Algoritmi	★★☆☆☆
Structuri de date	★★☆☆☆
Originalitate	★☆☆☆☆
Cunoștințe	★★☆☆☆
Implementare	★☆☆☆☆

http://www.ginfo.ro/revista/12_7/surse/conducte.cpp

P060220: Dune

Această problemă este, de fapt, o "schimbare la față" a problemei *Matrice* de la prima rundă a ediției a treia a concursului de programare *Bursele Agora*. Puteți găsi o scurtă prezentare a soluției în numărul 12/1 (ianuarie 2002) al *GInfo* (**www.ginfo.ro/revista/12_1**). Vom prezenta din nou soluția ținând cont de noul enunț.

Pentru rezolvarea problemei vom folosi metoda programării dinamice. Considerăm matricea de intrare X. Se observă că există cel mult 64 de modalități de a plasa bombe (mai exact colțurile din stânga-sus ale secțiunilor pătratice afectate de acestea) pe o linie a matricei. Valoarea 64 se obține datorită faptului că secțiunile trebuie să fie incluse integral în matrice (bombele nu pot atinge zone din afara *Câmpiei Arakeenului*), deci colțul din stânga-sus al unui pătrat poate fi plasat numai pe una dintre primele șase coloane. De fapt, acest număr poate fi redus la 37, deoarece unele posibilități de plasare sunt inutile.

De exemplu, se observă că patru bombe sunt suficiente pentru a acoperi integral o linie, deci nu are sens să plasăm cinci bombe pe o linie, deoarece cel puțin una dintre ele va

1. *Cormen T. H., Leiserson C. E., Rivest R. L., Introducere în Algortmi, Computer Libris Agora, Cluj-Napoca, 2000*

afecta poziții atinse și de alte bombe, deci bomba poate fi folosită în alte zone, valoarea diferenței dintre numărul *harkonnenilor* și cel al *atreizilor* uciși rămânând aceeași. De asemenea, nu are rost să plasăm colțurile din stânga-sus a zonelor afectate de trei bombe în trei coloane consecutive, deoarece pozițiile afectate de bomba din mijloc, sunt afectate de celelalte două bombe.

Presupunem că știm, pentru o anumită linie, sumele maxime care pot fi obținute dacă plasăm 0, 1, 2, ..., K bombe pe primele i linii, folosind pe linia i o anumită configurație. Apare întrebarea dacă putem determina aceleași informații pentru linia $i + 1$.

Răspunsul este afirmativ și ne oferă rezolvarea problemei. Construim (cel puțin teoretic) tabloul tridimensional $M[1 .. N - 1, 0 .. K, 1 .. 37]$. Un element $M[i, j, l]$ reprezintă suma maximă obținută dacă plasăm j pătrate pe primele i linii, folosind configurația l pe linia i.

Elementele matricei $M[i + 1]$ pot fi determinate folosind matricea $M[i]$. Pentru a calcula un element al matricei $M[i + 1]$ avem nevoie de următoarele informații:

- suma valorilor de pe linia $i + 2$ corespunzătoare unor zone afectate de bombe care vor fi plasate pe linia $i + 1$, folosind configurația l;
- numărul $nr[l]$ al bombelor care sunt folosite pentru obținerea configurației l;
- suma valorilor de pe linia $i + 1$ care corespund unor zone afectate dacă se folosește configurația l, dar nu și folosind o configurație c (determinăm valorile sumei pentru orice configurație c);
- valoarea maximă din tabloul M pentru fiecare configurație c plasată pe linia i care conține $j - nr[l]$ pătrate.

Programul este construit astfel:

- configurațiile posibile de pe o linie sunt păstrate în matricea de constante p, iar numărul de pătrate folosit pentru obținerea fiecărei configurații este păstrat în vectorul nr;
- tabloul tridimensional M este simulat cu ajutorul a două matrice A și B (adică $A = M[i]$, se calculează $B = M[i + 1]$, apoi A devine B etc.)
- pentru fiecare rând unde se pot plasa bombe:
  - se calculează suma adăugată pe rândul respectiv dacă plasăm o anumită configurație, în funcție de configurația plasată pe rândul anterior;
  - se calculează suma adăugată pe rândul următor dacă plasăm o anumită configurație pe rândul curent;
  - se inițializează matricea B;
  - se realizează calculul valorilor din B (datorită faptului că operațiile din această zonă sunt executate de foarte multe ori, codul trebuie optimizat cât mai mult posibil;
  - A devine B.

Soluția este dată de elementul maxim obținut în final în matricea A.

Analiza complexității

Citirea unei linii a matricei care reprezintă *Câmpia Arakeenului* se realizează în timp constant deoarece matricea are întotdeauna șapte coloane. Datorită faptului că trebuie citite n linii, ordinul de complexitate al operației de citire este $O(1) \cdot O(n) = O(n)$.

Matricele A și B au, fiecare, k linii și 37 de coloane. Operația de determinare a elementelor matricei B, folosind matricea A, are ordinul de complexitate $O(k)$, deoarece numărul operațiilor efectuate pentru determinarea uneia dintre linii este $O(37) \cdot O(37) \cdot O(7) = 37 \cdot O(1) \cdot 37 \cdot O(1) \cdot 7 \cdot O(1) = 9583 \cdot O(1) = O(1)$. Pentru fiecare dintre cele n linii ale matricei care caracterizează câmpia, se vor calcula elementele matricei B, deci ordinul de complexitate a operației de determinare a valorilor finale ale elementelor matricei B este $O(n) \cdot O(k) = O(n \cdot k)$.

Pentru afișarea diferenței maxime trebuie parcursă matricea finală, deci această operație are ordinul de complexitate $O(k)$.

În concluzie, algoritmul de rezolvare a acestei probleme are ordinul de complexitate $O(n) + O(n \cdot k) + O(k) = O(n \cdot k)$.

Dificultate 43

Algoritmi	★★★☆☆
Structuri de date	★☆☆☆☆
Originalitate	★★★☆☆
Cunoștințe	★★☆☆☆
Implementare	★☆☆☆☆

http://www.ginfo.ro/revista/12_7/surse/dune.cpp

P060221: Frăția Inelului

Vom construi un graf orientat ale cărui noduri reprezintă reperele și ale cărui arce reprezintă drumurile care leagă două repere. Sensul unui arc va fi de la reperul de la care pornește drumul la reperul la care ajunge drumul. Datorită faptului că nu se poate ajunge de la un reper la unul anterior, deducem că graful este aciclic.

Acum, problema se reduce la determinarea numărului de drumuri din acest graf de la nodul care reprezintă reperul `VALCEAUADESPICATA` la nodul care reprezintă reperul `MUNTELEOSANDEI`.

Pentru simplitate vom codifica reperele prin numere cuprinse între 0 și $N - 1$, unde N reprezintă numărul total al reperelor.

Pentru a determina numărul drumurilor dintre două noduri ale unui graf orientat vom folosi metoda programării dinamice. Vom construi un șir care reprezintă numărul de posibilități de a ajunge la un anumit reper (numărul de drumuri care pornesc din `VALCEAUADESPICATA` și ajung la reperul respectiv).

Pentru un anumit nod vom putea determina numărul de posibilități dacă se cunoaște numărul de posibilități în care se poate ajunge la fiecare dintre predecesorii săi din graf. Dacă toate aceste informații sunt cunoscute, atunci numărul posibilităților de a ajunge la acest reper va fi dat de suma numerelor posibilităților de a ajunge la reperele corespunzătoare predecesorilor.

Inițial considerăm că există o singură posibilitate de a ajunge la VALCEAUADESPICATA. Datorită aciclicității grafului suntem siguri că, la fiecare pas, vom determina cel puțin un nod pentru care cunoaștem valorile corespunzătoare tuturor predecesorilor. Așadar, la fiecare pas, vom verifica pentru toate nodurile pentru care nu cunoaștem numărul posibilităților, dacă sunt cunoscute numerele posibilităților corespunzătoare predecesorilor. Pentru fiecare nod pentru care sunt cunoscute aceste informații vom calcula valoarea corespunzătoare. Datorită faptului că la fiecare pas vom determina cel puțin un astfel de nod, suntem siguri că algoritmul se va termina după un număr finit de pași (cel mult N - 1).

În final, vom afișa valoarea corespunzătoare nodului care reprezintă MUNTELEOSANDEI.

Analiza complexității

Datele de intrare constau în drumurile dintre repere, deci ordinul de complexitate al operației de citire a acestora este $O(M)$.

Pentru fiecare reper citit se determină codul acestuia, deci ordinul de complexitate al operației de determinare a codurilor este $O(M) \cdot O(N) = O(M \cdot N)$ datorită faptului că acest cod este căutat într-un șir care conține cel mult N elemente.

Crearea listelor predecesorilor se realizează pe parcursul citirii datelor de intrare, operația având ordinul de complexitate $O(M)$.

Stabilirea faptului că există o posibilitate de a se ajunge în VALCEAUADESPICATA implică determinarea codului acestui reper, deci are ordinul de complexitate $O(N)$.

La fiecare pas al algoritmului vom verifica, pentru fiecare nod, dacă au fost determinate informațiile pentru predecesori. Pentru că un nod poate avea, teoretic, până la N - 1 predecesori, operațiile efectuate la fiecare pas au ordinul de complexitate $O(N) \cdot O(N) = O(N^2)$. Deoarece sunt efectuați cel mult N - 1 pași, operația de determinare a numărului de posibilități de a ajunge la MUNTELEOSANDEI are ordinul de complexitate $O(N) \cdot O(N^2) = O(N^3)$.

Scrierea datelor de ieșire implică determinarea codului pentru MUNTELEOSANDEI, deci are ordinul de complexitate $O(N)$.

În concluzie, algoritmul de rezolvare a acestei probleme are ordinul de complexitate $O(M) + O(M \cdot N) + O(M) + O(N) + O(N^3) + O(N) = O(N^3 + M \cdot N) = O(N \cdot (N^2 + M))$.

Dificultate 33

Algoritmi	★★☆☆☆
Structuri de date	★☆☆☆☆
Originalitate	★☆☆☆☆
Cunoștințe	★★☆☆☆
Implementare	★☆☆☆☆

http://www.ginfo.ro/revista/12_7/surse/lords.cpp

P060222: Litere

Problema se reduce la determinarea unui cuvânt care a fost codificat folosind transformarea *Burrows-Wheeler* dacă se cunoaște rezultatul codificării. Modul în care se realizează codificarea este prezentat în enunțul problemei.

Decodificarea cuvântului se realizează folosind următorul algoritm:

- se determină frecvența de apariție a literelor, ceea ce permite obținerea primei coloane a matricei (prima coloană conține întotdeauna literele ordonate lexicografic).
- se construiește un șir de corespondențe A între literele de pe prima coloană și cele de pe ultima, ținând cont de faptul că celei de-a i-a apariții a caracterului c de pe prima coloană îi corespunde cea de-a i-a apariție a caracterului c de pe ultima.
- pentru determinarea cuvântului se scriu literele corespunzătoare pozițiilor $A[1]$, $A[A[1]]$, $A[A[A[1]]]$ etc; pentru aceasta se folosește un indice i care, după fiecare pas va primi valoarea $A[i]$.

Acest algoritm a fost demonstrat matematic de către cei doi cercetători a căror nume a fost dat acestei metode.

Analiza complexității

Datele de intrare constau în citirea șirului literelor de pe ultima coloană; deoarece avem N litere, ordinul de complexitate al operației este $O(N)$.

Determinarea frecvențelor aparițiilor literelor se realizează în timp *liniar* prin simpla parcurgere a șirului care conține literele.

Construirea șirului de corespondențe se realizează tot în timp *liniar* deoarece la fiecare pas este realizată câte o corespondență.

Pentru afișarea datelor de ieșire (a cuvântului căutat) se realizează o simplă parcurgere a șirului de corespondențe, deci această operație are ordinul de complexitate $O(N)$.

În concluzie, algoritmul de rezolvare a acestei probleme are ordinul de complexitate $O(N) + O(N) + O(N) + O(N) = O(N)$.

Dificultate 28

Algoritmi	★★☆☆☆
Structuri de date	★☆☆☆☆
Originalitate	★☆☆☆☆
Cunoștințe	★★★☆☆
Implementare	★☆☆☆☆

http://www.ginfo.ro/revista/12_7/surse/litere.cpp

P060223: Zăvor

În cele ce urmează vom folosi următoarele denumiri pentru a ne referi la anumite stări ale automatului:

- *stare inițială* (stare de bază) - starea în care se află automatul la început sau după resetare (prima stare);
- *stare finală* (stare de deschidere) - starea în care se află automatul atunci când se deschide cifrul;

- *stare intermediară* - una dintre stările în care se află automatul pe parcursul introducerii parolei corecte (una dintre cele N - 1 stări corespunzătoare primelor N - 1 cifre ale parolei);
- *stare de eroare* - starea în care se ajunge dacă se recepționează un semnal incorect (a (N + 2) - a stare a automatului).

Vom descrie în continuare cele patru situații în care putem detecta că cipul nu funcționează corect:
- cipul poate fi deschis și în una dintre stările intermediare, nu numai în starea finală;
- există posibilitatea de a "sări" peste anumite stări intermediare;
- există posibilitatea de a ajunge într-o anumită stare în mai multe moduri;
- există posibilitatea de a reveni din starea de eroare în una dintre celelalte stări.

În cele ce urmează vom explica pe larg caracteristicile fiecărei situații și vom arăta modul în care poate fi găsită o modalitate alternativă de a deschide cifrul dacă a fost detectată o astfel de situație.

Pentru prima situație vom folosi cifrele din parola corectă și, la fiecare pas, vom verifica dacă una dintre stările intermediare i este stare de deschidere.

Parola alternativă va fi formată din cifrele de pe pozițiile 1, 2, ..., i ale parolei corecte.

Pentru a doua situație vom lua în considerare toate perechile i, j ($i < j$) și vom verifica dacă cifrul se deschide folosind cifrele de pe pozițiile 1, 2, ..., i, j + 1, ..., n, unde n este lungimea parolei corecte.

Parola alternativă va fi formată din cifrele de pe aceste poziții.

Pentru a treia situație vom lua în considerare toate tripletele i, j, k și vom verifica dacă din starea i se poate ajunge în starea j folosind semnalul k, unde semnalul k nu este cifra care duce la trecerea din starea i în starea i + 1.

Parola alternativă va fi formată din cifrele de pe pozițiile 1, 2, ..., i ale parolei corecte, semnalul k și cifrele de pe pozițiile j, j + 1, ..., n ale parolei corecte.

Pentru a patra situație vom lua toate perechile i, j și vom verifica dacă din starea de eroare putem ajunge în starea i folosind semnalul j. Pentru a ajunge în starea de eroare vom "trimite" la început un semnal z diferit de prima cifră a parolei.

Parola alternativă va fi dată de semnalul z, semnalul j și cifrele de pe pozițiile i, i + 1, ..., n ale parolei corecte.

Dacă nu s-a detectat nici o parolă alternativă, aceasta este scrisă în fișierul de ieșire.

În caz contrar, în fișierul de ieșire se va scrie șirul de caractere `CORECT`.

Analiza complexității

Trebuie remarcat faptul că operațiile efectuate de bibliotecă se realizează în timp ***constant***.

Datele de intrare constau în cele n cifre ale parolei corecte, deci citirea acestora are ordinul de complexitate $O(n)$.

Pentru prima situație vom trece pe rând dintr-o stare în alta (pornim din starea inițială și parcurgem stările intermediare) și vom testa la fiecare pas dacă s-a ajuns într-o stare de deschidere. Așadar, ordinul de complexitate al acestei operații este $O(n)$.

Pentru cea de-a doua situație există $n \cdot (n - 1) / 2$ perechi (i, j) pe care le vom lua în considerare. Pentru fiecare astfel de pereche vom parcurge stările intermediare corespunzătoare (vor exista cel mult n - 1 astfel de stări), deci ordinul de complexitate al operațiilor efectuate pentru fiecare pereche este $O(n)$. Așadar, pentru cea de-a doua situație ordinul de complexitate al tuturor operațiilor este $O(n^2) \cdot O(n) = O(n^3)$.

Pentru cea de-a treia situație există n^2 perechi (i, j) și $9 \cdot n^2$ triplete (i, j, k). Pentru fiecare triplet vom parcurge stările intermediare corespunzătoare care vor fi în număr de cel mult $2 \cdot n + 1$, deci ordinul de complexitate al operațiilor efectuate pentru fiecare triplet este $O(n)$. Așadar, pentru cea de-a treia situație ordinul de complexitate al tuturor operațiilor este $O(9) \cdot O(n^2) \cdot O(n) = 9 \cdot O(1) \cdot O(n^2) \cdot O(n) = O(1) \cdot O(n^2) \cdot O(n) = O(n^3)$.

Pentru cea de-a patra situație există $10 \cdot n$ perechi (i, j) și pentru fiecare dintre acestea vom parcurge stările intermediare corespunzătoare (cel mult n + 1 astfel de stări), deci ordinul de complexitate al operațiilor efectuate pentru fiecare pereche este $O(n)$. Așadar, pentru cea de-a patra situație ordinul de complexitate al tuturor operațiilor este $O(10) \cdot O(n) \cdot O(n) = 10 \cdot O(1) \cdot O(n) \cdot O(n) = O(1) \cdot O(n) \cdot O(n) = O(n^2)$.

În momentul în care detectăm o posibilitate alternativă de deschidere a cifrului, va trebui să construim noua parolă. Operația se realizează în timp ***liniar*** deoarece noua parolă va conține cel mult $2 \cdot n + 1$ cifre.

În final, vom scrie parola găsită sau cuvântul `CORECT`, operație realizabilă în timp ***liniar*** (dacă a fost găsită o parolă alternativă) sau ***constant*** (dacă nu a fost găsită o astfel de parolă).

În concluzie, algoritmul de rezolvare a acestei probleme are ordinul de complexitate $O(n) + O(n) + O(n^3) + O(n^3) + O(n^2) + O(n) + O(n) = O(n^3)$.

Dificultate 32

Algoritmi	★☆☆☆☆
Structuri de date	★☆☆☆☆
Originalitate	★☆☆☆☆
Cunoștințe	★★★☆☆
Implementare	★☆☆☆☆

http://www.ginfo.ro/revista/12_7/surse/zavor.cpp

Bursele Agora

Vă prezentăm în continuare enunțurile problemelor propuse spre rezolvare la primele patru etape ale ediției 2002/2003 a concursului de programare Bursele Agora.

P070207: Găuri de vierme

În anul 2457 cercetătorii din *Federația Planetelor Unite* au descoperit secretul găurilor de vierme. Astfel ei au reușit să construiască un dispozitiv de creare a unor găuri de vierme artificiale. Din nefericire proiectul nu este încă bine pus la punct; din aceste motive nu se poate călători decât într-un singur sens printr-o gaură de vierme artificială. Astfel o gaură de vierme are o intrare într-un anumit sistem stelar și o ieșire aflată într-un alt sistem stelar (care se poate afla la milioane de ani lumină distanță).

Construirea găurilor de vierme artificiale necesită folosirea unei cantități uriașe de energie, motiv pentru care *Federația Planetelor Unite* poate construi doar N astfel de scurtături prin spațiu. De aceea s-au ales cele mai importante N sisteme stelare din *Federație* și s-a hotărât că în fiecare astfel de sistem va exista o intrare a unei găuri de vierme și o ieșire a alteia.

Spionii din *Imperiul Galactic*, cel mai important inamic al *Federației*, au aflat de acest proiect și care sunt cele N sisteme stelare alese pentru acest experiment. *Consiliul Imperial* și-a dat seama că este de o importanță vitală să se știe care este numărul de posibilități de creare a găurilor de vierme, motiv pentru care vă cere să determinați această valoare.

Date de intrare

Fișierul de intrare **WORMHOLE.IN** conține numărul N al sistemelor stelare alese.

Date de ieșire

Fișierul de ieșire **WORMHOLE.OUT** trebuie să conțină numărul posibilităților de amplasare a găurilor de vierme astfel încât în fiecare sistem stelar să existe exact o intrare și exact o ieșire a două găuri de vierme diferite.

Restricții

- $2 \le N \le 255$.

Exemple

WORMHOLE.IN
3

WORMHOLE.OUT
2

WORMHOLE.IN
4

WORMHOLE.OUT
9

Timp de execuție: 1 secundă/test

P070208: Matrice

Se consideră o matrice A de dimensiune $M \times N$ ale cărei elemente au valoarea 0. Se aleg K cvadruple de numere x_1, y_1, x_2, y_2 cu $x_1 \le x_2$ și $y_1 \le y_2$. Pentru fiecare astfel de cvadruplu valoarea tuturor elementelor A_{ij} ($x_1 \le j \le x_2$, $y_1 \le i \le y_2$) crește cu 1.

Dându-se o matrice A, să se determine un număr cât mai mic de cvadruple care să ducă la obținerea matricei A, pornind de la o matrice care are toate elementele nule.

Date de intrare

Prima linie a fișierului de intrare **MATRIX.IN** conține numărul de linii M și numărul de coloane N ale matricei, separate printr-un singur spațiu.

Fiecare dintre următoarele M linii va conține N numere naturale, separate prin spații; al j-lea număr de pe a i-a dintre aceste N linii reprezintă valoarea elementului A_{ij}.

Date de ieșire

Prima linie a fișierului de ieșire **MATRIX.OUT** va conține numărul K al cvadruplelor folosite.

Fiecare dintre următoarele K linii va conține cele patru numere din componența unui cvadruplu, în ordinea x_1, y_1, x_2, y_2, separate prin spații.

Restricții și precizări

- matricea A va putea fi obținută întotdeauna folosind cel mult 1000 de cvadrupli;
- $1 \le M, N \le 100$;
- $0 \le A_{ij} \le 255$.

Modalitatea de acordare a punctajului

Pentru fiecare test, se vor putea obține cel mult X puncte.

Concurenții care vor găsi cea mai mică valoare K_{min} prin care se poate obține matricea A, vor primi X puncte pentru testul respectiv.

Ceilalți concurenți, care au reușit să obțină matricea A folosind $K > K_{min}$ cvadruple, vor obține $X \cdot \frac{K_{min}}{K}$ puncte pentru testul respectiv. Această valoare va fi aproximată cu două zecimale exacte.

Punctajul final va fi obținut prin adunarea punctajelor de la fiecare test și rotunjirea acestuia la cel mai apropiat număr întreg.

Dacă cvadruplele din fișierul de ieșire nu duc la obținerea matricei A sau programul nu respectă limita de timp admisă, concurenții nu vor primi nici un punct pentru testul respectiv.

De exmplu, dacă pentru un test se pot obține cel mult 5 puncte, cel mai bun rezultat obținut de un concurent constă în 7 cvadruple, iar dacă un alt concurent obține o soluție cu 12 cvadruple, atunci, pentru testul respectiv, punctajul concurentului va fi de $5 \cdot \frac{7}{12} \approx 2.92$ puncte.

Exemplu

MATRIX.IN

```
5 6
1 2 2 1 1 0
1 3 3 2 2 0
1 3 3 2 2 0
0 2 2 3 3 1
0 1 1 2 2 1
```

MATRIX.OUT

```
4
2 2 5 4
1 1 3 3
2 1 5 5
4 4 6 5
```

Timp de execuție: 1 secundă/test

P070209: Ecuații

Se consideră N variabile care pot avea valori numere reale și M ecuații în care pot apărea cele N variabile, operatorii de adunare ('+'), scădere ('-') și înmulțire ('*'), constante întregi cuprinse între -10000 și 10000 și, o singură dată, semnul de egalitate ('=').

Va trebui să determinați valori pentru cele N variabile astfel încât toate cele M egalități (ecuații) să fie respectate.

De exemplu, pentru variabilele `a`, `b`, `c` și `d` și ecuațiile:

```
a+b+c+d=18
a*b+c*d=43
a+b=41-c*d
a+b=-1219+c*d*d*c
```

valorile pot fi `a=2`, `b=4`, `c=5` și `d=7`. Soluția nu este unică; o altă posibilitate este `a=4`, `b=2`, `c=7` și `d=5`; mai există și alte soluții.

Se garantează faptul că există întotdeauna cel puțin o posibilitate de a atribui valori celor N variabile astfel încât toate cele M egalități să fie respectate.

În cele M ecuații nu apar paranteze, iar operatorii de înmulțire au prioritate față de cei de scădere și adunare. Aceștia din urmă au aceeași prioritate, deci operațiile de adunare și scădere sunt efectuate de la stânga la dreapta.

Date de intrare

Aceasta este o problemă cu *fișierele de intrare deschise*. În pagina **http://www.ginfo.ro/concurs/runda03/index.shtml**, pagină dedicată problemei sunt disponibile *link*-uri spre zece fișiere de intrare. Numele acestora au forma **EQXX.IN**, unde **XX** ia valori între **01** și **10** și reprezintă numărul testului.

Fiecare dintre cele 10 fișiere de intrare va conține, pe prima linie, numărul N al variabilelor. Pe fiecare dintre următoarele N linii se află identificatorul uneia dintre variabile. Identificatorii sunt formați din cel puțin una și cel mult opt litere ale alfabetului englezesc. Următoarea linie conține numărul M al ecuațiilor, iar pe fiecare dintre următoarele M linii apare una dintre ecuații, în formatul specificat anterior.

Date de ieșire

Pentru această problemă nu va trebui să trimiteți un program care să o rezolve, ci doar cele 10 fișiere de ieșire corecte. Acestea vor fi denumite **EQXX.OUT**, unde **XX** ia valori între **01** și **10** și reprezintă numărul testului.

Prima linie a fișierului de ieșire va conține textul `EQUATIONS - TEST #XX`, unde `XX` reprezintă numărul testului. Fiecare dintre următoarele N linii va conține un șir de tipul `variabilă = valoare`, unde `variabilă` este una dintre cele N variabile din fișierul de intrare, iar `valoare` este valoarea atribuită acesteia. Valorile vor fi scrise cu opt zecimale exacte. Variabilele vor apărea în fișierul de ieșire în aceeași ordine în care apar în fișierul de intrare corespunzător. Fiecare variabilă va apărea exact o dată.

Exemplu

Vom considera că acest exemplu reprezintă testul 00.

EQ00.IN

```
4
b
c
a
d
4
a*b+c*d=43
a+b=41-c*d
a+b=-1219+c*c*d*d
a+b+c+d=18
```

```
EQ00.OUT
EQUATIONS - TEST #00
b=2.00000000
c=5.00000000
a=4.00000000
d=7.00000000
```

Modalitatea de acordare a punctajului

Pentru fiecare fișier de ieșire corect veți obține 10 puncte. Punctajul maxim care poate fi obținut (dacă toate cele 10 fișiere de ieșire sunt corecte) este de 100 de punte.

Trimiterea soluțiilor

Arhiva pe care o veți trimite se va numi **YYYYYR03.ZIP**, **YYYYYR03.RAR** sau **YYYYYR03.ACE** (în funcție de programul de arhivare pe care îl folosiți), unde **YYYYY** este codul dumneavoastră de identificare (*ID*).

Ea nu va conține fișiere *batch* (**YYYYYR03.BAT**) sau fișiere sursă (**YYYYYR03.PAS**, **YYYYYR03.CPP** sau **YYYYYR03.C**). În arhivă se vor afla cele 10 fișiere de ieșire corecte, denumirea lor fiind **EQXX.OUT**, unde **XX** reprezintă numărul testului (cuprins între **01** și **10**).

P070210: Șah

Considerăm poziția finală a jocului de șah în care pe tablă au rămas regina albă, regele alb și regele negru.

Vom identifica regina albă prin litera *Q*, regele alb prin *K* și regele negru prin *k*.

Tabla de șah este formată din 64 de pătrățele dispuse într-o matrice cu opt linii și opt coloane. Coloanele matricei sunt identificate prin primele opt litere ale alfabetului (*a* - *h*), iar liniile prin primele opt numere naturale strict pozitive (1 - 8).

Poziția unei piese este dată de litera corespunzătoare coloanei și numărul corespunzător liniei. De exemplu, în figura 1 regina albă se află în poziția *a*2, regele alb în poziția *b*3, iar regele negru în poziția *g*4.

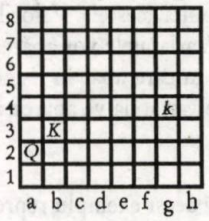


Figura 1

Definim aria de acțiune a reginei albe ca fiind totalitatea pătrățelelor de pe aceeași linie, coloană sau diagonală în care nu se află piese albe și între un astfel de pătrățel și regină nu se află nici o altă piesă. Pătrățelul în care se află regina nu face parte din aria sa de acțiune. În figura 2 este ilustrată aria de acțiune a reginei albe la un moment dat.

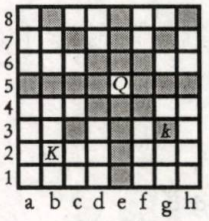


Figura 2

Definim aria de acțiune a unui rege ca fiind totalitatea pătrățelelor alăturate (pe linie, coloană sau diagonală) care nu sunt ocupate de piese de aceeași culoare. În figura 3 este ilustrată aria de acțiune a regelui alb într-o anumită situație.

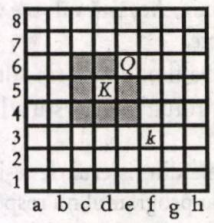


Figura 3

Jucătorii vor muta alternativ; considerăm că, la început jucătorul cu piesele albe este la mutare.

La un moment dat jucătorul, cu piesele albe poate muta fie regele, fie regina. Regele alb poate fi mutat în oricare dintre pătrățelele din aria sa de acțiune dacă pătrățelul respectiv nu se află în aria de acțiune a regelui negru.

Regina albă poate fi mutată în oricare dintre pătrățelele din aria sa de acțiune, chiar dacă intră în aria de acțiune a regelui negru.

Regele negru poate fi mutat în oricare dintre pătrățelele din aria sa de acțiune dacă, după mutare, nu ajunge în aria de acțiune a reginei albe sau a regelui alb.

Dacă o piesă ajunge în poziția ocupată de o altă piesă, atunci această a doua piesă este capturată și eliminată de pe tablă.

Se observă că, datorită regulilor anterioare, cei doi regi nu pot ajunge în poziții alăturate și singura piesă care poate fi capturată este regina albă.

Trebuie observat faptul că regele negru nu poate ajunge lângă regele alb prin capturarea reginei deoarece, chiar dacă momentan poziția respectivă nu este în aria de acțiune a regelui alb (respectând definiția anterioară), după mutare această poziție intră în aria de acțiune datorită eliminării reginei. De exemplu, în situația din figura 4 regina nu poate fi capturată.

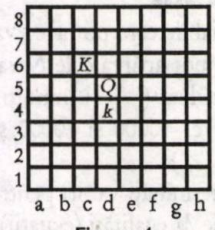


Figura 4

Se știe că într-o situație finală de acest tip jucătorul alb nu poate pierde; mai mult, el are șanse foarte mari de câștig.

Jucătorul negru pierde partida dacă se află în aria de acțiune a reginei albe și nu poate muta în nici unul dintre

pătrățelele din aria sa de acțiune. Această situație se numește *mat*.

Partida se termină la egalitate dacă regina albă este capturată sau se ajunge în situația de *pat*. Această situație apare atunci când regele negru nu se află în aria de acțiune a reginei albe, dar nu poate efectua nici o mutare validă. O astfel de situație este ilustrată în figura 5.

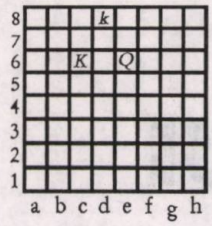


Figura 5

Veți juca cu piesele albe (regele și regina) și va trebui să câștigați partida folosind un număr limitat de mutări.

Mutările adversarului (regele negru) vor fi efectuate de o bibliotecă externă.

Bibliotecile Pascal și C/C++

Pentru a putea folosi biblioteca va trebui să includeți în programul dumneavoastră instrucțiunea **uses** `chess;` pentru limbajul *Pascal* și `#include "chess.h"` pentru limbajul *C/C++*. În continuare sunt prezentate sintaxele funcțiilor și procedurilor incluse în biblioteci:

- **procedure** `Init;`
  void `Init()`
  - trebuie apelată la începutul jocului și este folosită de către bibliotecă pentru inițializarea tablei;
  - întrerupe execuția programului dacă este apelată a doua oară.
- **function** `MovesLeft:Integer;`
  int `MovesLeft()`
  - returnează un număr întreg care reprezintă numărul de mutări pe care le mai aveți la dispoziție pentru a câștiga.
- **function** `GetWhiteQueenPosition:`**String**`;`
  char* `GetWhiteQueenPosition()`
  - returnează un șir care conține două caractere, primul fiind o literă care reprezintă coloana pe care se află regina albă în acel moment, iar al doilea un număr care reprezintă linia.
- **function** `GetWhiteKingPosition:`**String**`;`
  char* `GetWhiteKingPosition()`
  - returnează poziția curentă a regelui alb.
- **function** `GetBlackKingPosition:`**String**`;`
  char* `GetBlackKingPosition()`
  - returnează poziția curentă a regelui negru.
- **procedure** `MoveWhiteQueen(position:`**String**`);`
  void `MoveWhiteQueen(`**char***` position)`
  - mută regina albă în poziția indicată de șirul `position`;
  - întrerupe execuția programului dacă:
    - șirul de caractere nu reprezintă o poziție validă;
    - mutarea este nepermisă;
    - este a doua mutare consecutivă a jucătorului alb;
    - s-a epuizat numărul de mutări admis și partida s-a încheiat;
    - se ajunge în situație de *pat*;
    - se ajunge în situație de *mat* (singurul caz în care obțineți puncte pentru testul respectiv).
- **procedure** `MoveWhiteKing(position:`**String**`);`
  void `MoveWhiteKing(`**char***` position)`
  - mută regele alb în poziția indicată de șirul `position`;
  - întrerupe execuția programului dacă este verificată una dintre condițiile prezentate în cadrul procedurii anterioare.
- **procedure** `MoveBlackKing;`
  void `MoveWhiteKing()`
  - indică bibliotecii că se așteaptă mutarea regelui negru;
  - întrerupe execuția programului dacă:
    - este a doua mutare consecutivă a regelui negru;
    - prin mutare poate fi capturată regina albă;
  - noua poziție a regelui negru trebuie determinată printr-un apel al funcției `GetBlackKingPosition`.

Toate funcțiile și procedurile (cu excepția procedurii `Init`) întrerup execuția programului dacă sunt apelate înaintea inițializării tablei.

Programul vostru nu va citi datele din nici un fișier de intrare și nu va scrie date în nici un fișier de ieșire.

Exemplu de utilizare corectă a bibliotecii

În figura 6 este ilustrată poziția inițială:

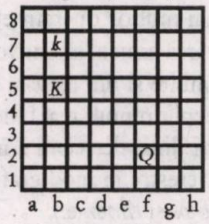


Figura 6

Subprogram apelat	*Valoare returnată*
`Init`	-
`MovesLeft`	12
`GetWhiteQueenPosition`	f2
`GetWhiteKingPosition`	b5
`GetBlackKingPosition`	b7
`MoveWhiteQueen(c5)`	-
`MoveBlackKing`	-
`GetBlackKingPosition`	a7
`MoveWhiteQueen(c7)`	-
`MoveBlackKing`	-
`GetBlackKingPosition`	a8
`MoveWhiteQueen(d7)`	-
`MoveBlackKing`	-
`GetBlackKingPosition`	b8
`MoveWhiteKing(b6)`	-
`MoveBlackKing`	-
`GetBlackKingPosition`	a8
`MoveWhiteQueen(b7)`	-

Timp de execuție: 1 secundă/test

Simularea PARALELISMULUI

Emilian-Marius Bold

În cadrul acestui articol vom prezenta o modalitate de simulare a paralelismului. Abordarea este exemplificată pentru simularea unor algoritmi paraleli simpli de sortare.

De multe ori suntem puși în situația de a realiza un program cât mai simplu, care să "pară" că folosește metode sofisticate cum ar fi paralelismul. Probabil că mulți dintre cei care citesc acest articol au încercat, cel puțin odată în perioada de inițiere în programare, să simuleze diferite tehnici superioare (sisteme de ferestre în mod grafic, efectuarea în paralel a mai multor operații etc). Acest lucru nu este inutil - nu încearcă nimeni să reinventeze roata - deoarece se pot învăța multe și din această tentativă. (Autorul recunoaște că și-a explicat mult mai bine lucrul cu pointeri, liste și alte structuri dinamice când a încercat să construiască un sistem de ferestre în mod grafic cu o simulare de *callback* gen *Windows* și *timer*-e.).

Un astfel de program simplu pe care am vrut să-l fac - a se citi trebuia să-l fac - avea scopul de a arăta cu cât este mai puțin bună metoda ***quicksort*** în comparație cu celelalte metode de sortare. Deci, avem mai mulți algoritmi de sortare care primesc același șir (de fapt mai multe șiruri care inițial conțin aceleași numere) și trebuie ca sortarea să se facă sub ochii utilizatorului.

Rularea în paralel a algoritmilor se putea realiza folosind posibilitățile ***multitasking*** ale sistemului de operare folosit, însă scopul declarat a fost de a scrie un program la nivel de liceu care să poată face aceste lucruri. Vom discuta în continuare doar la nivel de pseudocod, deci limbajul de programare în sine nu are nici o importanță.

Posibilitatea *multitasking* fiind eliminată, am urmat ideea ca fiecare procedură (corespunzătoare câte unui algoritm) să fie apelată de un număr de ori până se ajunge la sortarea șirului; la fiecare apel procedura trebuie să ruleze doar un "pas" din algoritm. Această idee poate fi transpusă astfel:

```
[1]
01: cât timp nu s-au sortat toate șirurile execută
02: apelează QuickSort
03: apelează BubbleSort
...
xx: sfârșit cât timp
```

Din nou subliniez faptul că un apel al unei proceduri nu determină sortarea imediată a șirului. La un apel procedura execută un număr de instrucțiuni, urmând ca la următorul apel să execute instrucțiunile care urmează în mod logic.

Încă de la început apare o idee evidentă: pentru ca aceasta să fie o simulare corectă a paralelismului trebuie ca fiecare procedură să aibă ca număr maxim de instrucțiuni o constantă.

Deoarece fiecare procedură are un timp de rulare $T(1)$ și evident și o complexitate $O(1)$ este ca și când s-ar oferi acesteia o cotă dintr-un procesor virtual.

Singurul lucru care deranjează într-o simulare este posibilitatea ca una dintre proceduri să acapareze firul de execuție. Tocmai de aceea instrucțiunile ciclice reprezintă un "pericol" ridicat. Ceea ce dorim să evităm sunt de fapt acele "salturi înapoi" - înțelegem printr-un salt înapoi un salt la o instrucțiune peste care am trecut deja, mai exact saltul la o instrucțiune care are o adresă mai mică decât adresa instrucțiunii curente (presupunând că adresele cresc) - și ne folosim astfel de operații decizionale (dacă). Evident că toată această transformare a operațiilor ciclice în operații decizionale poate să necesite variabile suplimentare.

În practică, procedurile se vor executa destul de repede astfel încât, pentru a putea observa modificările efectuate de algoritm, ar trebui utilizat câte un apel gen ***delay***.

De la prima implementare se observă o altă necesitate: trebuie să existe pentru fiecare algoritm o procedură de inițializare pe lângă cea de simulare deoarece procedura de simulare nu se poate baza pe datele pe care i le transmitem. Evident, codul de inițializare putea fi inclus în procedura

de simulare, urmând ca noi să transmitem un parametru special care să indice dacă este sau nu primul apel al procedurii.

Să luăm spre exemplu următorul algoritm de sortare:

```
[2]
1: repetă
2:    sch=fals
3:    pentru i=1, n-1 execută
4:       dacă a[i]>a[i+1] atunci
5:          sch=adevărat
6:          apelează Schimbă a[i],a[i+1]
7:       sfârșit dacă
8:    sfârșit pentru
9: până când not sch
```

Avem de-a face cu o variantă a algoritmului *Bubblesort*. Pentru a avea, după cum am mai spus, un număr fix de instrucțiuni executate la un apel al procedurii trebuie să simulăm instrucțiunile ciclice și eventualele apeluri recursive.

Procedura START se deduce din pseudocodul algoritmului, dacă ne amintim că instrucțiunea **pentru** este echivalentă cu:

```
[3]
1: i=1
2: dacă i<=n atunci
3:    execută corpul instrucțiunii "pentru"
4:    i=i+1
5:    salt la linia 2
6: sfârșit dacă
```

Structurile repetitive sunt de fapt înlocuite de structuri alternative, eventual adăugând variabile suplimentare - sună cunoscut?

Astfel, o implementare ar putea fi:

```
[4]
START:
1: sch=fals
2: i=1
BSORT:
01: dacă i<n atunci
02:    dacă a[i]>a[i+1] atunci
03:       sch=adevărat
04:       apelează Schimbă a[i],a[i+1]
05:    sfârșit dacă
06: sfârșit dacă
07: i=i+1
08: dacă i>=n atunci
09:    dacă sch atunci
10:       sch=fals
11:       i=1
12:    altfel
13:       încheiere algoritm - sortat!
14:    sfârșit dacă
15: sfârșit dacă
```

În procedura BSORT presupunem că suntem deja în interiorul ciclului **repetă**. La linia 1 se verifică dacă suntem în interiorul ciclului **pentru**, iar dacă da, urmează instrucțiunile care le observăm și în liniile 4-7 [2].

După ce mărim *i* la linia 7 putem avea două cazuri: *i* este mai mic decât *n* și, deci, încă nu am ieșit din ciclul **pentru** sau este egal cu *n*, ceea ce înseamnă că trebuie să vedem dacă vom ieși și din **repetă**. Cazul în care *i* este mai mare decât *n* apare doar când *n* este mai mic decât 2; este un caz particular de mic interes.

O dată încheiat ciclul **pentru** (linia 8) verificăm dacă au fost efectuate schimbări sau nu. Dacă au fost, ne pregătim pentru un nou ciclu **pentru**; observați asemănarea dintre liniile 10-11 și liniile 1-2 din START. În momentul în care ajungem la linia 13 înseamnă că nu au fost efectuate schimbări, șirul este ordonat și putem declara că algoritmul de sortare s-a încheiat. Această linie nu este echivalentă cu o simplă ieșire din procedură - în practică am folosit o variabilă suplimentară pentru a determina dacă șirul este sortat sau nu; variabila devenea fals în START și adevărat abia în acest moment.

Ar fi un exercițiu interesant pentru cititori realizarea în același mod al algoritmului SelectSort.

Algoritmul prezentat a fost destul de ușor, deoarece nu apare recurență în algoritmul inițial. În continuare vom observa că recurența modifică în mare măsură modul de simulare.

Vom încerca algoritmul Quicksort. Varianta sa în pseudocod este:

```
[5]
QUICKSORT(primul, ultimul)
01: i=primul
02: j=ultimul
03: temp=a[(i+1)/2]
04: repetă
05:    cât timp a[i]<temp execută
06:       i=i+1
07:    sfârșit cât timp
08:    cât timp a[i]>temp execută
09:       j=j-1
10:    sfârșit cât timp
11:    dacă i<j atunci
12:       apelează Schimbă a[i],a[j]
13:    sfârșit dacă
14:    dacă i<=j atunci
15:       j=j-1
16:       i=i+1
17:    sfârșit dacă
18: până când i>=j
19: dacă primul<j atunci
20:    apelează QUICKSORT primul,j
21: sfârșit dacă
22: dacă i<ultimul atunci
23:    apelează QUICKSORT i,ultimul
24: sfârșit dacă
```

În continuare va trebui să simulăm stiva, iar procedura va lucra mereu cu elementele din vârful acesteia. Altfel spus, variabilele `i`, `j` etc. vor fi variabile din vârful stivei simulate. Structura stivei va include `primul`, `ultimul`, `i`, `j`, `temp`, `pas2`. Procedura de inițializare devine:

```
[6]
START
0: A=vidă
1: A<=(1,n,0)
```

Prin `A<=(X,Y,Z)` înțelegem aici introducerea în stiva `A` a unui bloc care are `primul=X`, `ultimul=Y` și `pas2=Z` iar celelalte variabile sunt nedefinite (deocamdată). În același mod se definește `x<=A` ca reprezentând scoaterea unui nod din `A` și depunerea în `x`.

Elementele nedefinite ca urmare a procedurii de inițializare vor fi definite de către procedura de simulare în sine. Elementul `pas2` este o variabilă suplimentară care devine 1 doar atunci când s-a realizat definirea elementelor despre care vorbim. Variabila `temp` nu putea fi inițializată în `START`, deoarece se presupune acces la șirul `a` și credem că este mai firesc ca procedura de simulare să fie singura cu acces la acest șir. Această procedură devine:

```
[7]
QSORT
01: dacă pas2=0 atunci
02:   i=primul
03:   j=ultimul
04:   temp=a[(i+j)/2]
05:   pas2=1
06: sfârșit dacă
07: dacă a[i]<temp atunci
08:   i=i+1
09: sfârșit dacă
10: dacă a[i]>temp atunci
11:   j=j-1
12: sfârșit dacă
13: dacă (a[i]<temp) sau (a[i]>temp) atunci
14:   ieșire QSORT
15: sfârșit dacă
16: dacă i<j atunci
17:   apelează schimbă a[i], a[j]
18: sfârșit dacă
19: dacă i<=j atunci
20:   j=j-1
21:   i=i+1
22: sfârșit dacă
23: dacă i>=j atunci
24:   k=0
25:   dacă primul<j atunci
26:     A<=(primul,j,0)
27:     primul=j;
28:     k=1
29:   sfârșit dacă
30:   dacă i<ultimul atunci
31:     A<=(i,ultimul,0)
32:     ultimul=i;
33:     k=1
34:   sfârșit dacă
35:   dacă k=0 atunci
36:     x <= A
37:     dacă A=vidă atunci
38:       încheiere algoritm - sortat!
39:     sfârșit dacă
40:   sfârșit dacă
41: sfârșit dacă
```

Este foarte important faptul că toate variabilele din algoritmul de mai sus, cu excepția lui `x` și `k`, sunt de fapt cele din vârful stivei. Mai mult, `x` nici nu trebuie să fie o variabilă propriu-zisă - este suficientă ștergerea nodului din vârful stivei (cu actualizările de rigoare).

La linia 1 se testează dacă au fost inițializate toate variabilele.

Liniile 7-9, 10-12 corespund secvențelor **cât timp** din algoritmul original, doar că se execută câte un singur pas.

Liniile 13-15 verifică dacă putem trece la instrucțiunea de la linia 11 [5]. Pentru a respecta complexitatea $O(1)$ a versiunii pseudo-paralele, ciclurile **cât timp** nu se pot reproduce exact și de aceea apelăm la aceasta metodă. Observați, totuși, că la un apel al procedurii se pot executa atât linia 8 cât și 11 - dorim ca la un apel să se poată executa cât mai multe instrucțiuni.

Liniile 16-21 sunt aceleași cu 11-17 din `[5]`, iar linia 22 verifică dacă am ajuns la sfârșitul ciclului **repetă** din `[5]`.

Liniile 25-29, 30-34 corespund liniilor 19-21, 22-24 din `[5]`. Deoarece simulăm stiva, **apelează** `QSORT x, y` devine `A<=(x,y,0)`. Liniile 28 și 33 au un rol aparte: `k=1` marchează faptul că s-au introdus elemente în stivă, iar prin operațiile `primul=j` și `ultimul=i` ne asigurăm că la o viitoare apelare a procedurii pentru acest nod al stivei nu vom mai introduce din nou alte noduri.

Linia 35 testează dacă `k=0`. Aceasta înseamnă că nu s-au introdus noduri, deci putem șterge din stivă fără prea mari probleme. Sortarea se încheie când `A` este vidă.

Se poate observa că în cazul unui bloc din stivă avem două cazuri:

- dacă acesta nu determină crearea altor noduri, el se va șterge în același moment în care vom ajunge la pasul 16;
- dacă acesta determină crearea altor blocuri în stivă el se va șterge în urma unui apel ulterior al procedurii, după ce toate blocurile care se vor crea după ce a fost creat el se vor șterge. Acest mod de a șterge blocul în urma unei apelări practic vide a procedurii (căci ultima apelare nu face altceva decat să șteargă nodul) poate fi înlocuit cu o altă metodă de a crea noi blocuri: unul din cele două posibile noi blocuri ar putea înlocui conținutul blocului curent. Se poate ajunge astfel și la o utilizare redusă a stivei. (Din păcate în program nu am folosit această metodă deși pare foarte bună!).

Emilian-Marius Bold este student la Universitatea de Vest din Timișoara. Poate fi contactat prin e-mail la **fierarul@info.uvt.ro** .

Cluj-Napoca, 22 iulie 2002

ADMITERE UBB 2002

Militon Frențiu

În urma corectării lucrărilor scrise la proba de Informatică, dar și a examenelor de la disciplina "Algoritmică și programare" cu studenții anului I, am observat că absolvenții claselor de informatică nu sunt deprinși să folosească subprograme. De aceea, în urmă cu doi ani am publicat [1] un articol despre importanța conceptului de subalgoritm în programare. În acest articol veți putea citi despre examenul de admitere din acest an.

În enunțul problemei date în acest an la admitere, am menționat expres subprogramele care trebuie să apară în programul cerut. Cu toate acestea, mai mult de jumătate din candidații care au ales să dea examen la disciplina "Informatică" nu au avut în programul lor toate subprogramele menționate, iar dintre cei care le-au avut, o bună parte nu au respectat cerințele din enunțul dat. Cea mai frecventă greșeală, pentru cei care au scris subprogramele cerute, constă în lipsa parametrilor formali, deci folosirea variabilelor globale. Acest fapt dovedește o înțelegere greșită a conceptului de subprogram și a importanței subprogramelor în programare.

Pentru a putea fi refolosite în orice context subprogramele nu trebuie să folosească variabile globale. În consecință, variabilele care marchează datele presupuse cunoscute în subproblema corespunzătoare și cele care marchează rezultatele obținute trebuie să apară în lista parametrilor formali.

O altă greșeală, prezentă și la candidații care au luat note mari (care au gândit bine, dar n-au redat în scris nimic despre modul în care au gândit), este lipsa oricăror explicații referitoare la program. Din cei 183 de candidați doar trei au dat explicațiile necesare. O parte au dat unele explicații prin comentarii prezente în textul programului. Dar nici importanța comentariilor nu este cunoscută absolvenților de liceu. Doar 28 de lucrări au avut cel puțin un rând de comentarii, dar numai 7 dintre acestea au fost acceptabile, nici una perfectă.

Menționăm, fără a le analiza, alte neajunsuri, destul de frecvente: scriere necaligrafică, dezordonată, cu multe ștersături, cu multe greșeli, algoritmi incorecți, programe neclare sau cu artificii în dauna clarității, algoritmi complicați în locul unora destul de simpli, variabile inutile, construirea unor tablouri care nu sunt necesare în program. Convinși de faptul că acești candidați sunt elevii cei mai bine pregătiți la informatică, putem trage o concluzie despre nivelul de pregătire al absolvenților de liceu.

Problemele date în acest an au fost următoarele:

1. Descrieți în *pseudocod* algoritmi pentru rezolvarea următoarelor probleme:
   a. Determinați dacă o matrice pătratică care are elemente distincte poate fi transformată prin schimbări de linii într-una în care elementul minim al fiecărei coloane se află pe diagonala principală. Dacă răspunsul este da, rearanjați matricea.
   b. Ordonați crescător elementele de pe diagonala principală a unei matrice pătratice, prin schimbări de linii și coloane.
   c. Se dă un tablou unidimensional X cu componente numere reale. Eliminați din X toate componentele negative.

2. Scrieți un program *Pascal* care citește numerele naturale $n_1, n_2, \ldots, n_k$ și tipărește cea mai lungă secvență $n_s, n_{s+1}, \ldots, n_d$, cu $1 \le s \le d \le k$, care conține numai numere prime. Programul trebuie să conțină:
   - o procedură care citește numerele date;
   - o funcție care verifică dacă un număr natural este prim;
   - o procedură care determină indicii s și d, $1 \le s \le d \le k$, cu proprietatea că $n_s, n_{s+1}, \ldots, n_d$ sunt numere prime;
   - o procedură care tipărește numerele $n_s, n_{s+1}, \ldots, n_d$.

Diferența esențială între un algoritm și subalgoritmul corespunzător aceleiași probleme constă în faptul că algoritmul citește datele de intrare și tipărește rezultatele obținute, în timp ce subalgoritmul primește prin parametri datele de intrare și transmite, tot prin parametri, rezultatele obținute. Calculele efectuate sunt aceleași în ambele situații.

Deci, pentru aceeași problemă *P*, în care datele de intrare sunt depuse în *X*, iar rezultatele se depun în *R*, următorul algoritm rezolvă problema *P*:

```
Algoritmul AP este:
   Citește X;
   Cheamă SP(X, R);
   Tipărește R
sfârșit algoritm
```

unde *SP* este subalgoritmul corespunzător problemei *P*. El primește din locul apelului datele de intrare *X* și retransmite rezultatele *R*. În programe complicate, datele *X* pot fi rezultatul unor calcule anterioare (deci nu pot fi citite), iar rezultatele obținute și transmise la locul apelului, pot fi doar rezultate intermediare utile în calculele ulterioare; nefiind rezultate finale, ele nu vor fi *tipărite*.

Întrucât orice problemă poate apărea ca o subproblemă a unei probleme mai complicate, se recomandă scrierea unui subalgoritm pentru rezolvarea ei. De aceea, dăm în continuare subalgoritmii ceruți la punctele *a*, *b*, respectiv *c*. Pentru candidați au fost acceptați fie algoritmii fie subalgoritmii corespunzători.

a.

```
Funcția Minim(A,n,j) este:
{Găsește linia ce conține valoarea minimă de pe a j-a coloană a matricei A}
r:=1;
Pentru i:=2; n execută
   Dacă A[i,j]<A[r,j]
      atunci r:=i
   sfârșit dacă
sfârșit pentru
Minim:=r
sfârșit Minim;

Subalgoritmul InterSchimbLin(A,n,i,k)
                                    este:
{Interschimbă liniile i și k în matricea A}
Pentru j:=1;n execută
   temp:=A[i,j];
   A[i,j]:=A[k,j];
   A[k,j]:=temp
sfârșit pentru
sfârșit InterSchimbLin

Subalgoritmul Rearanjez(A,n,b) este:
{Efectuează rearanjarea cerută. Dacă aceasta este posibilă
b=1, altfel b=0}
b:=1; {Ipoteza "se poate face rearanjarea"}
i:=1;
Cât timp i≤n și b=1 execută
   k:=Minim(A,n,i);
   Dacă i>k
      atunci b:=0
      {Rearanjarea nu se poate efectua!}
   altfel Dacă i<k atunci
            Cheamă InterSchimbLin(A,i,k);
          sfârșit dacă
          i:=i+1;
  sfârșit dacă
sfârșit cât timp
sfârșit Rearanjez
```

b. Pentru a descrie subalgoritmul de ordonare cerut avem nevoie de doi subalgoritmi auxiliari pentru interschimbarea a două linii, respectiv coloane.

Subalgoritmul `InterSchimbLin(A,n,i,k)` este dat deja la punctul a.

```
Subalgoritmul InterSchimbCol(A,n,j,k)
                                    este:
{Interschimbă coloanele j și k în matricea A}
Pentru i:=1; n execută
   temp:=A[i,j];
   A[i,j]:=A[i,k];
   A[i,k]:=temp
sfârșit pentru
sfârșit InterSchimbCol

Subalgoritmul Ordonare (A,n) este:
{Ordonează crescător elementele de pe diagonala matricei
A prin interschimbări de linii și coloane}
Repetă
   Ordonat:=1;
   {Ipoteza "este ordinea dorită"}
   Pentru i:=1; n-1 execută
      Dacă A[i,i]>A[i+1,i+1] atunci
      {Elementele nu sunt în ordine, deci interschimbare}
         Cheamă InterSchimbLin(A,n,i,i+1);
         Cheamă InterSchimbCol(A,n,i,i+1);
         Ordonat:=0
      sfârșit dacă
pânacând Ordonat=1 sfârșit repetă
sfârșit Ordonare
```

c. Problema cere eliminarea componentelor negative dintr-un tablou `X`, nu copierea celorlalte componente într-un alt tablou. Deci, rezultatul obținut va fi tot `X`, specificarea subalgoritmului fiind:

```
DATE n, X; {n≥1, X fiind un tablou cu n componente
                numere întregi}
REZULTATE n, X; {X conține doar valorile nenegative din
                cele date inițial, iar n este numărul lor}
```

Mai subliniem că parcurgerea tabloului `X` nu poate fi făcută printr-un ciclu **Pentru** deoarece, eliminând o componentă x_i, să zicem x_2, pe această poziție vine următoarea valoare, x_3, care poate fi tot negativă și trebuie testată (ea numindu-se tot x_2), deci `i` nu poate fi încă modificat, în acest caz trebuind să rămână 2. Aici nu se specifică ni-

mic despre păstrarea ordinii inițiale a elementelor care rămân în x. Dacă e permisă schimbarea acestei ordini (varianta aleasă în soluția dată mai jos), eliminarea va fi mai simplă, în locul unei valori negative putând aduce oricare altă valoare neprocesată. Subalgoritmul corespunzător este prezentat în continuare:

```
Subalgoritmul ELIMIN(n,X) este:
{Elimină din tabloul X cu n componente, valorile negative}
i:=1;
Cât timp i≤n execută
   Dacă x_i<0
      atunci x_i:=x_n; n:=n-1
           {x_i este șters, fiind înlocuit cu x_n}
      altfel i:=i+1
   sfârșit dacă
sfârșit cât timp
sfârșit ELIMIN
```

Dacă nu este permisă schimbarea ordinii elementelor (ceea ce ar fi trebuit menționat expres în textul problemei), atunci în locul atribuirii $x_i := x_n$ erau necesare instrucțiuni care deplasau toate valorile din dreapta cu o poziție spre stânga, ceea ce se poate face prin instrucțiunile

```
Pentru p:=i+1; n execută
   x_{p-1}:=x_p
sfârșit pentru
```

Pentru rezolvarea problemei 2 propunem o abordare *bottom-up*, rezolvând separat subproblemele problemei date. Aceste subprobleme sunt: *citirea datelor, afișarea rezultatelor, verificarea primalității unui număr, stabilirea unei secvențe de numere prime începând cu un indice dat* și *determinarea celei mai lungi secvențe de numere prime dintre secvențele găsite.*

Având în vedere că citirea și afișarea necesită subalgoritmi simpli, iar stabilirea primalității se află în acest număr al revistei la pagina 41, în cele ce urmează vom prezenta succint subalgoritmii pentru stabilirea unei secvențe de numere prime începând cu un indice dat și determinarea celei mai lungi secvențe de numere prime dintre secvențele găsite.

Să observăm că dacă am găsit o secvență de numere vecine prime după care urmează un număr neprim, următoarea secvență va începe cu cel puțin o poziție spre dreapta. De aceea este necesar să parcurgem o singură dată vectorul N dat, și pentru aceasta avem nevoie de un indice i. În cazul în care N_i este prim, dorim să găsim cea mai lungă secvență de numere prime care începe pe poziția i, secvență precizată prin indicele j pe care se află ultimul număr prim. În continuare prezentăm acest subalgoritm:

```
Subalgoritmul SECVENTA(i,k,N,j) este:
{Pornim cu valoarea i care este indicele numărului prim
N_i. Obținem cea mai mare valoare j pentru care secvența
N_i, N_{i+1}, ..., N_j conține numai numere prime}
j:=i;
Cât timp (j≤k) și (N_j este prim) execută
   j:=j+1
sfârșit cât timp
j:=j-1;
sfârșit SECVENTA
```

Subalgoritmul LONGSECV trebuie să furnizeze indicii de la capetele celei mai lungi subsecvențe care conține doar numere prime. Vom avansa în șirul dat până când descoperim primul număr prim, moment în care apelăm subalgoritmul SECVENTA. La început, neavând nici o secvență de numere prime, valoarea inițială a variabilei d este 0, iar valoarea provizorie a variabilei s este 1. După revenirea din subalgoritmul SECVENTA, avem la dispoziție în j marginea din dreapta a secvenței de numere prime care începe pe poziția i. În concluzie, la pasul următor, dacă secvența curentă este mai lungă decât una găsită anterior, atribuim valorile i, j variabilor s, respectiv d.

```
Subalgoritmul LONGSECV(k,N,s,d) este:
{Se obțin indicii s și d pentru care secvența N[s..d] conține numai numere prime și este cea mai lungă}
i:=1
s:=1
d:=0 {codifică "Șirul nu conține numere prime"}
cât timp i<=k execută
   cât timp (i<=k) și (N[i] nu este prim) execută
      i:=i+1
   sfârșit cât timp
   Dacă (i<=k) și (N[i] este prim) atunci
      cheamă Secventa(i,k,N,j)
      dacă (j-i > d-s) atunci
   {s-a găsit o secvență mai lungă}
         s:=i
         d:=j
      sfârșit dacă
      i:=j+1
   sfârșit dacă
sfârșit cât timp
sfârșit LONGSECV
```

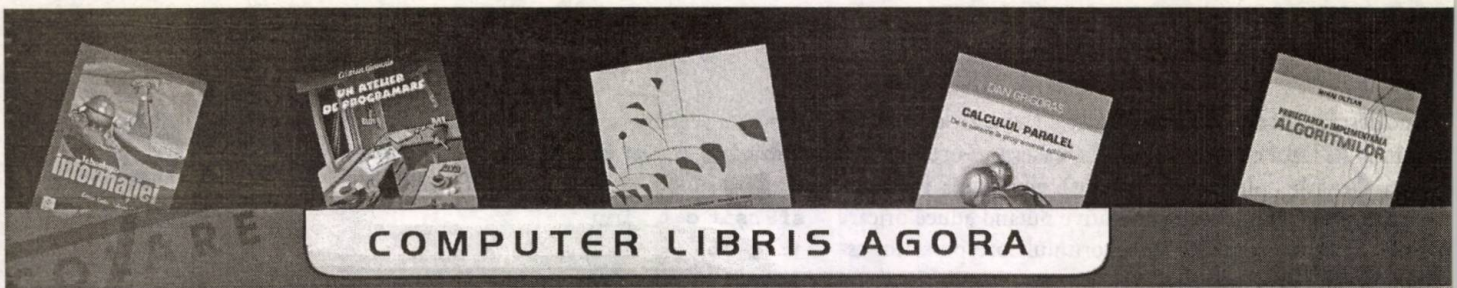
Celelalte explicații sunt date prin comentariile inserate în implementarea programului. Orice program trebuie să conțină un minim de comentarii explicative și trebuie să fie scris cu indentare pentru a mări claritatea textului. Textul sursă și versiunea integrală a acestui articol pot fi găsite la adresa:

http://www.cs.ubbcluj.ro/~mfrentiu/articole/adm2002.html

Bibliografie

1. M. Frențiu, *Conceptul de subalgoritm și importanța lui în programare*, GInfo 10/6.

Domnul prof. Militon Frențiu este cadru didactic la Universitatea "Babeș-Bolyai" și poate fi contactat prin e-mail la adresa **mfrentiu@cs.ubbcluj.ro.**



COMPUTER LIBRIS AGORA

Calculul paralel. De la sisteme la aplicații tratează, într-un mod natural și structurat, cele mai importante aspecte ale proiectării aplicațiilor paralele, sau, cu un termen mai nou, de mare performanță. Paralelismul, care înseamnă executarea mai multor operații simultan, este o caracteristică a tuturor calculatoarelor actuale, inclusiv a celor personale. De aceea, cunoașterea principiilor și legilor care îl guvernează, ca și a metodologiilor pentru proiectarea aplicațiilor cele mai performante este o necesitate, dacă se dorește exploatarea optimă a resurselor sistemelor de calcul. Parametrii de performanță și metodologiile de proiectare și execuție optimă reprezintă subiectele care guvernează discursul științific al cărții de față.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

EUREKA - rezolvitor de probleme

Marin Vlada

Numărul mare de produse software destinate rezolvării problemelor din diverse domenii este impresionant. Apariția acestora a fost posibilă ținând seama de rezultatele din domeniul calculatoarelor și de metodele și tehnicile noi realizate în fiecare domeniu. În acest articol vom prezenta unul dintre aceste produse, și anume rezolvitorul Eureka.

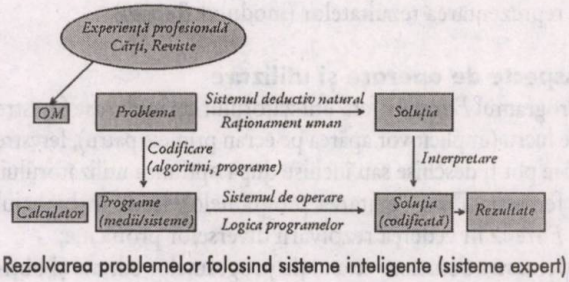
Cel care utilizează calculatorul este pus deseori în dificultate deoarece, uneori, trebuie să opteze pentru un produs sau altul. Alegerea depinde de diverși parametri, cum ar fi modul de a afla despre un anumit produs care rezolvă anumite probleme, dificultatea de utilizare, performanțele produsului, teoria care se află la baza elaborării sale, dificultatea de a-l obține și, nu în ultimul rând, prețul de achiziție. De exemplu, în domeniul matematicii există zeci de astfel de programe; de asemenea, pentru domeniul chimiei există foarte multe programe care rezolvă o mare diversitate de probleme. Există produse *software* care rezolvă anumite probleme la *nivel de expert*; este vorba de așa-numitele ***sisteme expert*** care sunt sisteme de programe care înglobează experiența unor specialiști și tehnicile cele mai avansate din domeniul respectiv; *inteligența artificială* are un rol important în conceperea și elaborarea acestor *sisteme expert*.

Rezolvarea problemelor

Utilizatorul care își desfășoară activitatea într-un anumit domeniu se confruntă cu diverse probleme pe care trebuie să le rezolve cu ajutorul calculatorului, ținând seama de complexitatea lor și de realizarea unei activități eficiente.

Sistemele de calcul clasice sunt au la bază funcționării lor principiul execuției *pas cu pas* (*step by step*), enunțat de *John von Neumann*.

Sistemele de operare și *limbajele de programare* au fost concepute și elaborate în conformitate cu acest principiu de bază. Din acest punct de vedere, în domeniul *procesării informației* nu s-au realizat progrese semnificative pentru *modelarea reprezentării informației* și pentru *tehnicile de prelucrare* ale acesteia. Prin inventarea microprocesorului și obținerea unor rezultate spectaculoase în domeniul *inteligenței artificiale*, s-a reușit conceperea și elaborarea de noi produse *hardware* și *software*. Acestea au schimbat tehnologia de prelucrare a informației prin *tehnologii de prelucrare* a cunoștințelor, tehnologii care se află la baza rezolvării problemelor.

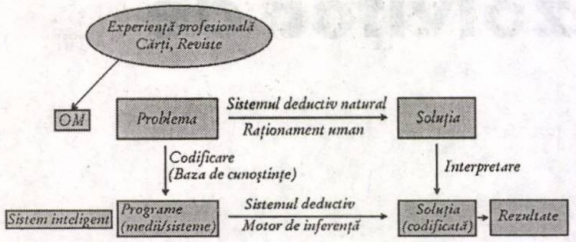


Rezolvarea problemelor folosind sisteme inteligente (sisteme expert)

Ca urmare a dezvoltării acestor tehnologii, s-au conceput și realizat *sisteme de calcul inteligente* care au facilitat elaborarea de *sisteme expert* care înglobează atât performanțele *hardware*, cât mai ales *procesarea cunoștințelor la nivel de expert* pentru diverse domenii. Astfel, au apărut sistemele inteligente (deoarece nu mai respectă principiul secvențialității specific modelului *von Neumann*, ci implementează *tehnici de calcul paralel/distribuit* și *motoare de inferențe*, în scopul simulării comportamentului inteligent al omului în rezolvarea problemelor).

Având în vedere complexitatea problemelor, utilizatorul este nevoit să studieze cazurile în care trebuie să decidă ce sistem de calcul va fi adecvat pentru problema în cauză, dar în același timp și să "intuiască" *modul de abordare a raționamentelor* care trebuie implementate pentru rezol-

varea problemei. În primul rând, el trebuie să se *informeze* despre existența *problemelor înrudite* cu cea în cauză, ce soluții au fost folosite și, în al doilea rând, să decidă ce sistem de calcul trebuie utilizat, precum și ce *produse software* pot fi folosite pentru rezolvarea cu calculatorul a problemei ce se studiază.



Rezolvarea problemelor folosind sisteme clasice (numerice)

Descrierea rezolvitorului Eureka

Programul *Eureka* este un mediu de rezolvare (*The Solver*) pentru:

- repezentarea grafică a funcțiilor reale de o singură variabilă (modulul *Graph*);
- tabelarea unei funcții reale de o singură variabilă (modulul *Graph*);
- calcule numerice pentru rezolvarea de probleme folosind metode și tehnici de analiză numerică (metode iterative, metode directe etc.) (modulele *Solve*, *Commands*);
- elaborarea de programe pentru rezolvarea problemelor (modulele *Edit*, *File*) folosind un limbaj de programare simplu, asemănător limbajului matematic;
- reprezentarea rezultatelor (modulul *Report*).

Aspecte de operare și utilizare

Programul *Eureka* oferă utilizatorului cel mult șase ferestre de lucru (implicit vor apărea pe ecran primele patru), ferestre care pot fi deschise sau închise după opțiunea utilizatorului:

- fereastra *Edit* - editarea programelor scrise în limbajul *Eureka* în vederea rezolvării diverselor probleme;
- fereastra *Solution* - execuția programelor editate și obținerea soluțiilor corespunzătoare;
- fereastra *Verify* - verificarea soluțiilor găsite pentru problema rezolvată;
- fereastra *Report* - prezentarea rezultatelor pe ecran;
- fereastra *Plot* - reprezentarea grafică a funcțiilor;
- fereastra *List* - lista tabelării unei funcții reale.

Fereastra activă este evidențiată printr-un *dublu-chenar* și este operațională, adică se pot efectua operații specifice ferestrei respective (de exemplu, pentru fereastra *Edit* sunt valabile operații de editare).

Limbajul Eureka

Limbajul *Eureka*, destinat codificării problemelor în vederea rezolvării lor, este foarte apropiat *limbajului matematic*. Este constituit din aspecte sintactice și semantice simple ale unui *limbaj de programare*. Sintaxa limbajului este simplă și comodă. Se utilizează un *alfabet minimal* și *construcții lexicale* simple.

Simboluri alfabetice utilizate special:

- caracterul `';'`, aflat la începutul unui rând, are semnificația *rând de comentariu* (instrucțiune de comentariu);
- caracterul `'='` reprezintă *operatorul de egalitate* (este folosit pentru scrierea de ecuații, sisteme);
- grupul de caractere `':='` reprezintă *operatorul de definire* (inițializare)
- caracterul `'$'` este utilizat pentru instrucțiuni de atribuire și operații de maxim și minim.

Expresiile matematice se construiesc cu operatorii: + (adunare), - (scădere), * (înmulțire), / (împărțire), ^ (exponent), e (notația științifică: `men = m · 10^n`).

Operatori relaționali utilizați sunt <, >, <= și >=.

Un *nume de variabilă* (*identificator*) este un șir de litere și cifre care semnifică un anumit aspect legat de rolul variabilei în descrierea problemei. Funcțiile oferite de *Eureka* sunt identificate printr-un *nume*, numit ***identificator***.

Exerciții

Să se calculeze valoarea integralei

$$v = \int_0^1 e^{-x^2} dx$$ (funcția lui *Gauss*).

Se selectează regimul *Edit* și se editează în fereastra *Edit* următoarele:

```
f(x) := exp(-x^2)
v = integ (f(x), x, 0, 1),
```

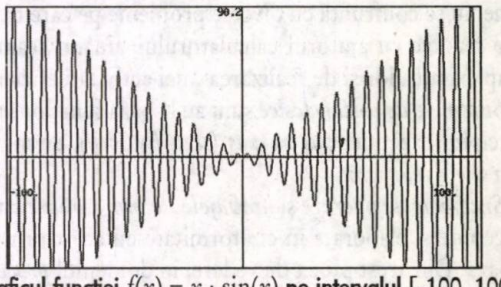
după care se acționează *Solve* pentru rezolvare. Rezultatul apare în fereastra *Solution*:

```
v = 0.74682413.
```

Să se reprezinte grafic funcția:

$$f : [-100, 100] \to R, f(x) = x \cdot \sin x.$$

Se va utiliza modulul *Graph*, unde se va selecta subfuncția *Function* pentru a se introduce expresia funcției f conform sintaxei limbajului *Eureka*, și anume, se tastează `f(x) = x * sin(x)`, după care se selectează subfuncția *Plot* pentru a se introduce capetele intervalului pe care se dorește reprezentarea grafică, și anume -100, apoi 100. Va apărea fereastra *Plot* cu reprezentarea punctuală a graficului. Pentru vizualizarea continuă a graficului se apasă tasta **F5** care realizează *Zoom*.



Graficul funcției $f(x) = x \cdot \sin(x)$ pe intervalul [-100, 100]

Domnul conf. dr. Marin Vlada este cadru didactic la Universitatea București și poate fi contactat prin e-mail la adresa **vlada@chem.unibuc.ro.**

TEOREME de programare

Clara Ionescu

În numărul precedent al GInfo am trecut în revistă clasa de probleme în care prelucrarea unui șir de date se concretizează în determinarea unui singur rezultat (o sumă, un produs, o reuniune etc.). Dacă într-o astfel de problemă rezultatul cerut trebuie determinat în prezența unei anumite proprietăți, este posibil ca algoritmul prezentat să prevadă multe operații inutile. Teorema următoare elimină acest neajuns.

Decizia

Să căutăm trăsătura comună a următoarelor probleme!

P5. Să se stabilească dacă un număr dat este prim sau nu!

P6. Având la dispoziție șirul cuvintelor care desemnează lunile anului, să se stabilească despre un cuvânt dat dacă reprezintă o lună din an sau nu!

P7. Măsurând pe durata lunii iunie temperatura apei *Mării Negre* în dreptul stațiunii *Mamaia*, să se stabilească dacă apa mării s-a încălzit încontinuu sau nu!

P8. Cunoscând mediile generale ale elevilor dintr-o clasă, să se stabilească dacă un anumit elev este premiant sau nu (se află sau nu printre elevii care au primele trei medii din clasă)!

Trăsătura comună a acestor probleme constă în felul în care algoritmul va furniza răspunsul. Soluția va fi dată sub forma unui mesaj prin care se confirmă sau se infirmă proprietatea cerută. Acest mesaj de multe ori poate fi simplu: `'DA'` sau `'NU'` și se afișează în funcție de valoarea unei variabile logice, stabilită în algoritm.

Totuși, printre enunțurile de mai sus există și diferențe. De exemplu, problema P5 cere verificarea unui singur număr. Evident, va trebui să studiem acest singur număr, astfel încât să putem decide dacă el este prim sau nu. Problema P6 cere verificarea existenței unei valori printre mai multe valori date. Deci studiem mai multe date, dar acest "studiu" se reduce la o simplă comparare a elementului curent din șirul dat cu valoarea căutată. În problema P7 verificăm o proprietate a unui întreg șir de date, deci trebuie să analizăm relația dintre valorile elementelor (ele trebuie să formeze un șir crescător). Problema P8 cere verificarea apartenenței unei valori date la o parte a unui șir dat.

Să rezolvăm problema P6 cu algoritmul prezentat în numărul precedent al revistei!

Începem cu faza preliminară în care stabilim variabilele și semnificațiile lor. Deocamdată ne "străduim" să respectăm descrierea algoritmului cunoscut:

n: întreg {*numărul elementelor șirului de prelucrat*}
x: tablou(1..n): tip element {*elementele șirului dat*}
S: logic {*rezultatul*}

În plus, vom folosi o funcție logică `P` în care vom verifica proprietatea cerută. Parametrul funcției `P` va fi un element al șirului, iar subalgoritmul de rezolvare a problemei are ca parametri șirul dat, dimensiunea acestuia și valoarea logică stabilită în funcție de existența sau absența proprietății căutate. În subalgoritmul următor am înlocuit identificatorul `S` cu `găsit`, pentru a-l face mai sugestiv.

```
subalgoritm Decizie(n,x,găsit):
  găsit ← fals
  pentru i ← 1, n execută:
    găsit ← găsit sau P(x[i])
  sfârșit pentru
sfârșit subalgoritm
```

Să observăm o proprietate importantă! Dacă la un moment dat valoarea variabilei `găsit` devine `adevărat`, valoarea ei nu se va mai schimba până la sfârșitul executării algoritmului. Respectiv, în cazul problemei P5, dacă am găsit un număr diferit de 1 și de numărul dat care divide numărul considerat, adică am aflat că numărul dat nu este prim, nu putem ajunge în final la concluzia că acesta de fapt este prim! Rezultă că toate operațiile efectuate după descoperirea unui divizor sunt inutile. Să modificăm, pe baza acestei observații, subalgoritmul `Decizie`!

În primul rând vom schimba structura repetitivă de tipul **pentru** într-o structură cu număr *necunoscut* de pași,

pentru a putea întrerupe executarea în momentul în care am obținut răspunsul la întrebare. Evident, în cazul problemei P6, dacă nici un element al șirului analizat nu coincide cu elementul căutat, se va parcurge întreg șirul.

```
subalgoritm Decizie1(n,x,găsit):
  i ← 1
  cât timp (i ≤ n) și nu (P(x[i])) execută:
    i ← i + 1
  sfârșit cât timp
  găsit ← i ≤ n
sfârșit subalgoritm
```

Să analizăm acum problemele din cealaltă categorie, mai exact, cele care seamănă cu problema P7. În această problemă fiecare element al șirului dat trebuie să aibă o aceeași proprietate, și anume, trebuie să aibă valoarea mai mare sau egală decât elementul precedent. Această caracteristică a șirului se poate formula și în felul următor: *în șir nu există nici un element care* ***nu*** *are proprietatea cerută.* Transformând subalgoritmul `Decizie1` prin aplicarea negației în două locuri, obținem algoritmul de rezolvare pentru această clasă de probleme. Deoarece semnificația deciziei se referă la toate elementele șirului, înlocuim variabila `găsit` cu `toate`.

```
subalgoritm Decizie2(n,x,toate):
  i ← 1
  cât timp (i ≤ n) și (P(x[i])) execută:
                          {am negat P(x[i])}
    i ← i + 1
  sfârșit cât timp
  toate ← i > n                    {am negat i ≤ n}
sfârșit subalgoritm
```

Probleme legate de implementare

Din nou vom intra în... laborator.

În momentul codificării subalgoritmilor `Decizie` pot apărea diverse probleme. În anumite limbaje de programare o expresie logică formată din subexpresii între care apar operatorii logici **și**, respectiv **sau**, se evaluează în întregime, dar există și limbaje în care, în funcție de operator și valoarea primei subexpresii, evaluarea se întrerupe după stabilirea valorii primei subexpresii. De exemplu, dacă valoarea de adevăr a primei subexpresii este `adevărat` și urmează operatorul **sau**, cea de-a doua subexpresie nu se mai evaluează, deoarece indiferent de valoarea acestuia, rezultatul final va fi `adevărat` (`adevărat` **sau** `adevărat` este `adevărat`, respectiv `adevărat` **sau** `fals` este tot `adevărat`). De asemenea, dacă valoarea de adevăr a primei subexpresii este `fals` și urmează operatorul **și**, cea de-a doua subexpresie nu se mai evaluează, deoarece indiferent de valoarea acesteia, rezultatul final va fi `fals` (`fals` **și** `adevărat` este `fals`, respectiv `fals` **și** `fals` este tot `fals`).

În cazul acelor limbaje de programare care, în schimb, nu întrerup evaluarea acestor expresii logice, sau dacă programatorul nu alege atent ordinea subexpresiilor în cazul primei categorii de limbaje de programare, codificarea deciziei, conform algoritmilor prezentați, pot genera erori neplăcute. Despre ce este vorba? Fie expresia logică din structura repetitivă **cât timp** din algoritmul `Decizie2`:

`(i ≤ n) și (P(x[i]))`.

Dacă `i > n`, în urma evaluării subexpresiei `i ≤ n` obținem `fals`, dar se evaluează și subexpresia `P(x[i])` în condițiile în care știm deja că `i > n`, deci se va verifica un element `x[i]` care nu există în șirul dat!

În concluzie, în implementarea subalgoritmilor de tipul celor prezentați trebuie să căutăm soluții prin care să evităm producerea unor erori în timpul executării sau obținerea unor rezultate incorecte. În cele ce urmează propunem câteva soluții pentru evitarea acestei "capcane".

a. putem crea un al $(n + 1)$-lea element fictiv;
b. putem opri reluarea executării nucleului structurii repetitive cu "un pas mai repede", urmând să analizăm motivele ieșirii din ciclu;
c. putem despărți subexpresiile, urmând ca pe cea de-a doua s-o evaluăm doar atunci când acest lucru este necesar;
d. să evităm folosirea, în cea de-a doua subexpresie, a valorii `x[i]`; acest lucru este posibil dacă introducem o variabilă logică pentru a reține valoarea de adevăr a subexpresiei `P(x[i])` de la pasul precedent, astfel permițând ieșirea din structura repetitivă în momentul în care avem confirmarea proprietății sau după parcurgerea întregului șir în cazul în care acest lucru nu a avut loc.

Să rezolvăm problema **P5** (numărul dat este prim sau nu?). Implementăm subalgoritmul `Decizie1` sub forma unei funcții, deoarece rezultatul este o singură valoare. Într-o primă abordare, pornim de la definiția numărului prim (un număr prim are exact doi divizori: 1 și el însuși). În concluzie, vom verifica dacă numărul dat are vreun divizor printre numerele 2, 3, ..., n-1.

```
function prim1(nr:Word):Boolean;          { P5 }
var diviz:Word;
begin
  diviz:=2;
  while (diviz <= nr - 1) and (nr mod diviz <> 0) do
    Inc(diviz);
  prim:=diviz > nr - 1
end;
```

Algoritmul poate fi îmbunătățit considerabil dacă tratăm separat cazul acelor numere care nu necesită nici o împărțire pentru stabilirea proprietății. Acestea sunt numărul 1 și numerele pare. În rest, proprietatea de divizibilitate o vom verifica doar cu numere impare, știind că un număr impar nu poate avea nici un divizor par. În plus, nu vom genera divizori mai mari decât rădăcina pătrată a numărului, deoarece divizorii posibili sunt mai mici decât această valoare.

```
function prim2(nr:Word):Boolean;          {P5}
var diviz:Word;
    p:Boolean;
    rad:Word;
begin
  if nr = 1 then p:=false
  else
    if not Odd(nr) then          {numere pare}
      p := nr = 2
           {singurul număr par prim este 2}
    else begin                   {numere impare}
      diviz := 3;
      rad := Trunc(Sqrt(nr));
      p := true;
      while (diviz <= rad) and p do
        if nr mod diviz = 0 then
          p := false          {diviz divide nr}
        else
          Inc(diviz,2)
    end;
  prim := p
end;
```

În această variantă a implementării am preferat să folosim variabila locală `p` în locul scrierii explicite a expresiei de verificare a divizibilității, pentru a mări lizibilitatea programului. Nu am putut folosi identificatorul `prim`, deoarece astfel în instrucțiunea **while** am fi avut un apel recursiv al funcției `prim`.

În continuare vom rezolva problema P6. Această problemă cere să stabilim dacă un cuvânt dat (`cuv`) reprezintă o lună din an sau nu. Presupunem că în vectorul de *string*-uri, denumit `luni`, având tipul `Tluni` avem păstrate cuvintele care reprezintă lunile anului. În funcția `Exista` variabila locală `gasit` primește valoarea `true` în urma depistării unui cuvânt care desemnează o lună care coincide cu șirul dat.

```
function Exista(luni:Tluni;cuv:string):Boolean;
var gasit:Boolean;
    i:Byte;
begin
  gasit := false;
  i := 1;
  while (i <= 12) and not gasit do
    if cuvant = luni[i] then
      gasit := true
    else
      Inc(i);
  Exista := gasit
end;
```

Problema P7 o vom rezolva implementând algoritmul `Decizie2`. Presupunem că tipul `tablou` este cunoscut. În această implementare în limbajul *Pascal* condiția din instrucțiunea **while** este "sigură", deoarece avansând cu `i` până inclusiv la valoarea `n - 1`, compararea lui `x[i]` cu `x[i+1]` la ultima iterație înseamnă compararea ultimelor două elemente din șir. Dacă am "reușit" să ajungem cu `i` la valoarea `n`, înseamnă că proprietatea este satisfăcută.

```
function Monoton(n:Byte;x:tablou):Boolean;
var i:Byte;
begin
  i := 1;
  while (i <= n - 1) and (x[i] <= x[i+1]) do
    Inc(i);
  Monoton:= i > n - 1
end;
```

În rezolvarea problemei P8 presupunem că în program s-au declarat tipurile și variabilele:

```
type elev = record
              nume:string;
              medie:Real
            end;
     elevi = array [1..30] of elev;
var clasa:elevi;
    nume:string;      {numele elevului cautat}
    n:Byte;
```

De asemenea, pentru a putea rezolva problema în mod eficient, presupunem că tabloul `clasa` este ordonat descrescător după `medie`. În continuare prezentăm funcția `premiant` cu care rezolvăm problema.

```
function premiant(nume:string;clasa:elevi):
                                        Boolean;
var i:Byte;
begin
  i := 1;
  while (i <= 3) and not (nume = clasa[i].nume) do
    Inc(i);
  premiant := i <= 3
end;
```

Evident, nu trebuie să "ne păzim" de eventuala depășire a dimensiunii șirului, deoarece într-o clasă de elevi cu siguranță vom avea mai mulți decât trei copii.

Concluzii

Dacă într-o problemă se cere doar verificarea unei anumite proprietăți, este recomandabil ca algoritmul să fie prevăzut cu posibilitatea opririi analizei imediat după ce proprietatea s-a stabilit. Dacă într-un șir dat se caută un element având o anumită proprietate, de asemenea este la limita minimei optimizări să nu căutăm elementul respectiv după ce l-am găsit! Evident, cu totul altfel se pune problema, dacă trebuie să găsim toate elementele având acea proprietate. În aceste probleme va trebui să parcurgem toate datele.

RECUNOAȘTEREA formelor

Radu-Daniel Vatavu

Serialul despre recunoașterea formelor continuă cu prezentarea unei tehnici des folosite în acest domeniu, și anume învățarea supravegheată. Vă vom prezenta câteva dintre regulile de bază, și anume regula celui mai apropiat vecin, cea a celor mai apropiați k vecini, cea a lui Bayes etc.

În procesul de învățare supravegheată se presupune faptul că există un set de învățare pe baza căruia se poate construi un model de clasificator care va fi folosit în procesul de recunoaștere. Fie F mulțimea formelor furnizate sistemului, C mulțimea claselor și S setul de învățare:

$$S = \{(x_i, c_j) / x_i \in F, c_j \in C\}$$

$$|F| = p, |C| = m,$$

unde p și m reprezintă numărul de elemente ale mulțimii F, respectiv C.

Mulțimea formelor F poate fi partiționată în m clase:

$$F = \{F_0, F_1, \ldots, F_{m-1}\}, F_i \cap F_j = \varnothing,$$

$$\forall i, j \in \{0,1,\ldots,m-1\}, i \neq j, F = \bigcup_{j=0}^{m-1} F_j .$$

O clasă F_i se numește *omogenă* dacă este îndeplinită condiția:

$$d(x_1, x_2) \leq d(x_1, y), d(x_1, x_2) \leq d(x_2, y)$$
$$\forall x_1, x_2 \in F_i, \forall y \notin F_i,$$

unde d desemnează o distanță.

Pentru formele din aceleași clase F_i se poate calcula o formă "medie" care reprezintă centrul de greutate al fiecărei clase:

$$m_i = \frac{1}{n_i} \sum_{j=0}^{n_i - 1} x_j^i$$

unde prin n_i s-a notat numărul formelor din clasa F_i, iar x_j^i este forma de indice j care aparține clasei i ($j = 0, \ldots, n_i - 1$).

Regula celui mai apropiat vecin

Cea mai intuitivă metodă de clasificare a unei noi forme x, plecând de la o mulțime de clase și o distanță d, este reprezentată de regula celui mai apropiat vecin (*NN - Nearest Neighbour*). Aceasta constă în calcularea distanțelor dintre forma x și fiecare formă x_i ($i = 0, 1, \ldots, p - 1$), care aparțin mulțimii F, luându-se următoarea decizie:

$$x \in F_i \Leftrightarrow \exists\, y \in F_i \text{ a.î. } \min\{d(x, x_i) / x_i \in F\} = d(x, y)$$

Deci, se va considera că noua formă aparține clasei din care face parte cel mai apropiat vecin, pentru o anumită distanță d.

Trebuie notat faptul că această regulă folosește ca informație privind clasificarea numai cel mai apropiat vecin, ignorând pur și simplu celelalte forme (respectiv distribuția lor în alte clase). Ca urmare, rezultatul obținut s-ar putea să nu fie întotdeauna corect, fie datorită prezenței unui anumit nivel de zgomot suprapus peste valorile caracteristicilor, fie datorită faptului că forma y a fost greșit clasificată. Unele dezavantaje prezentate de această regulă pot fi eliminate prin fundamentarea clasificării pe apartenența la clase a mai multor vecini.

Regula celor mai apropiați k vecini

Regula *KNN* (*K Nearest Neighbours*) ia în considerare cei mai apropiați k vecini ai formei x, decizia fundamentându-se astfel: forma x aparține clasei din care fac parte cei mai mulți dintre cei k vecini.

Numărul k trebuie ales astfel încât să fie suficient de mare pentru a minimiza probabilitatea unei clasificări greșite și suficient de mic (în raport cu numărul p al formelor) astfel încât cei k vecini să fie într-adevăr "aproape" de x pentru a asigura o estimare corectă a clasei.

Această metodă este des utilizată datorită avantajelor pe care le prezintă: nu sunt necesare informații privind probabilitatea de apartenență a unei forme la o anumită clasă, este ușor de implementat și prezintă o probabilitate mică de eroare.

În figura 1 este prezentată diferența dintre regulile *NN* și *KNN*, considerându-se un caz ipotetic de clasificare a unei forme necunoscute la una dintre cele două clase. S-a presupus că numărul caracteristicilor utilizate este $n = 2$ pentru a se facilita reprezentarea formelor în spațiul caracteristicilor.

Dezavantajul acestor două metode constă în faptul că, de fiecare dată când se dorește clasificarea unei noi forme,

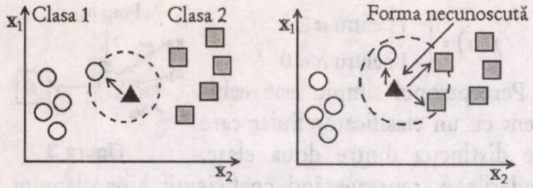


Figura 1: a) forma este atribuită clasei 1;
b) forma este atribuită clasei 2.

este necesară calcularea a *p* distanțe, ceea ce poate determina un timp de calcul ridicat.

Funcții discriminant

În spațiul caracteristicilor, fiecărei forme îi corespunde un punct având drept coordonate valorile celor *n* caracteristici:

$$x = (x_0, x_1, \text{K}\ , x_{n-1})$$

Alegerea corespunzătoare a caracteristicilor va determina reliefarea similarității dintre forme prin apropierea dintre punctele din spațiul caracteristicilor. După cum se observă în figura 1, formele din fiecare clasă sunt grupate. Acest lucru determină posibilitatea separării claselor printr-o curbă discriminant (sau o hipersuprafață discriminant într-un plan *n*-dimensional).

Dacă există hipersuprafețe discriminant care să separe planul caracteristicilor astfel încât formele care aparțin aceleiași clase să se găsească în aceeași regiune, atunci clasele se numesc ***separabile***. Dacă hipersuprafețele discriminant sunt hiperplane, atunci clasele se numesc ***liniar separabile***.

Dacă mulțimea claselor are cardinalul *m*, hipersuprafețele sunt definite de *m* funcții $g_i(x)$, i=0, 1, ..., m - 1 numite ***funcții discriminant***. Alegerea acestor funcții trebuie să se facă astfel încât să fie îndeplinită condiția:

$$g_i(x) > g_j(x),\ \forall j = 0, ..., m-1, j \neq i\ '$$

pentru orice formă *x* aparținând clasei *i*.

Ca urmare, decizia de apartenență a unei forme necunoscute *x* la o clasă se va fundamenta astfel:

$$x \in F_j \Leftrightarrow g_j(x) = \max\{g_i(x) / i = 0, ..., m-1\} \qquad (*).$$

Regiunile de decizie sunt separate de hipersuprafețele discriminant. Hipersuprafața de separație dintre clasele *i* și *j* este dată de ecuația:

$$g_{ij} = g_i(x) - g_j(x) = 0$$

și conține toate punctele (formele) *x* aflate la distanță egală față de clasele *i* și *j*.

În cazul în care clasele sunt liniar separabile, funcțiile discriminant vor fi funcții liniare de tipul:

$$g(x) = a_0x_0 + a_1x_1 + \text{K}\ + a_{n-1}x_{n-1} + a_n,$$
$$a_i \in \Re,\ x = (x_0, x_1, \text{K}\ , x_{n-1})$$

sau

$$g(x) = a \cdot x^T + a_n,\ a = (a_0, a_1, \text{K}\ , a_{n-1}),\ x^T = \begin{pmatrix} x_0 \\ x_1 \\ \text{M} \\ x_{n-1} \end{pmatrix}$$

Pentru a determina funcțiile discriminant g_i, se pot folosi vectorii formă medie ai fiecărei clase (m_i) în baza următoarei definiții:

$$g_i(x) = F\{d(x, m_i)\},\ i = 0, ..., m-1,$$

unde *d* este o distanță iar *F* este o funcție descrescătoare.

Ca urmare, forma *x* va aparține clasei pentru care distanța de la *x* la forma medie a clasei este minimă (asemănător cu regula *NN*).

Pentru cazul distanței euclidiene, distanța dintre forma *x* și forma medie m_i poate fi rescrisă:

$$d^2(x, m_i) = (x - m_i) \cdot (x - m_i)^T = (x - m_i) \cdot (x^T - m_i^T) =$$
$$= x \cdot x^T - x \cdot m_i^T - m_i \cdot x^T + m_i \cdot m_i^T =$$
$$= x \cdot x^T - 2 \cdot m_i \cdot x^T + m_i \cdot m_i^T =$$
$$= -2 \cdot \left[m_i \cdot x^T + 0.5 \cdot m_i \cdot m_i^T\right] + x \cdot x^T .$$

Având în vedere faptul că pentru un anumit *x*, factorul $x \cdot x^T$ este constant pentru fiecare i = 0, ..., m-1, funcțiile discriminant pot fi alese:

$$g_i(x) = m_i \cdot x^T + 0.5 \cdot m_i \cdot m_i^T .$$

Astfel, conform criteriului (*) definit anterior, forma necunoscută *x* va fi clasificată în clasa *j* pentru care $g_j(x)$ are valoarea maximă sau, altfel spus, distanța $d(x, m_j)$ are valoarea minimă.

Procesul de învățare constă în determinarea coeficienților funcțiilor discriminant $g_i(x)$.

Pe baza considerațiilor de mai sus se obține algoritmul de clasificare care este prezentat în continuare. Presupunem că funcțiile discriminant sunt deja calculate conform metodei prezentate, iar matricea *g*, de ordin $m \cdot (n+1)$, conține coeficienții. Mulțimea *C* a claselor este reprezentată de vectorul `clasa`. Forma necunoscută a fost notată cu *x*.

```
clasa_x ← -1
max ← 0
pentru i ← 0, m - 1 execută
   temp ← 0
   pentru j ← 0, n - 1 execută
      temp ← temp + g_ij * x_j
   sfârșit pentru
   temp ← temp + g_in
   dacă temp > max atunci
      max ← temp
      clasa_x ← j
   sfârșit dacă
sfârșit pentru
scrie clasa[clasa_x]
```

În figura 2 este prezentat un caz ipotetic care cuprinde trei clase și indică suprafețele de separație dintre acestea. Fiecare formă este descrisă de două caracteristici x_1 și x_2.

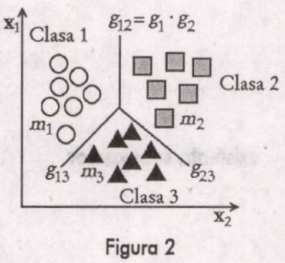


Figura 2

Dacă un astfel de clasificator, bazat pe distanța minimă față de centrul de greutate al fiecărei clase, oferă performanțele așteptate, nu există motive pentru utilizarea unor metode mai complicate (cum ar fi rețelele neuronale). Însă, există anumite situații în care rezultatul returnat de clasifi-

cator nu este cel corect. Câteva dintre acestea, precum și posibile soluții, sunt prezentate în tabelul 1. Au fost luate în considerare două clase și două caracteristici.

Rețele neuronale

După cum rezultă din tabelul 1, în anumite situații, suprafețele de separație dintre clase nu sunt liniare, ci devin chiar foarte complexe. În această situație, un clasificator de tipul celui prezentat mai sus nu poate oferi performanțele așteptate. În astfel de condiții, realizarea unui clasificator care să asigure rezultatele corespunzătoare se poate face apelând la rețelele neuronale. În continuare vor fi amintite modelele cunoscute ale *perceptronului* simplu și multistrat, prezentându-se și algoritmii de învățare pentru aceste modele.

Perceptronul este prezentat structural în figura 3, ieșirea sa y având valoarea:

$$y = f\left(\sum_{j=0}^{n-1} w_j \cdot x_j + w_n\right),$$

unde funcția de activare poate fi funcția prag simplă:

$$f(a) = \begin{cases} 1 \text{ pentru } a \geq 0 \\ -1 \text{ pentru } a < 0 \end{cases}.$$

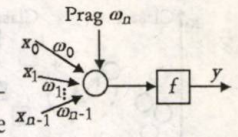


Figura 3

Perceptronul simplu este echivalent cu un clasificator liniar care face distincția dintre două clase, ponderile ω_j reprezentând coeficienții hiperplanului de separație. Astfel, forma x de intrare va fi clasificată în clasa 1 dacă $y = 1$ sau în clasa 2 dacă $y = -1$. Deci, pentru o clasificare corectă avem:

$$y \cdot z > 0, \quad \text{unde } z = \sum_{j=0}^{n-1} w_j \cdot x_j + w_n.$$

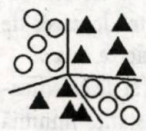
În cazul în care ieșirea perceptronului nu respectă valoarea clasei din setul de învățare, valorile ponderilor ω_j vor fi modificate corespunzător:

$$w_j = w_j + \alpha \cdot x_j, j = 0, \ldots, n-1,$$

unde α este o constantă de corecție pozitivă.

Algoritmul pentru învățarea perceptronului este prezentat în continuare:

Se aleg valori aleatoare mici pentru ω_j, j = 0, ..., n
Se alege o valoare pentru constanta α $(0 < \alpha \leq 1)$

Descriere Situație	Soluție posibilă
caracteristici alese necorespunzător	Un clasificator liniar nu va putea niciodată separa cele două clase. O soluție ar fi construirea unor caracteristici mai bune, însă această operație nu se realizează întotdeauna ușor. Dacă se cunosc anumite informații despre distribuția formelor în clase și diferite probabilități condiționate de apartenență se pot aplica alte metode (de exemplu, *regula lui Bayes*).
caracteristici corelate	Există posibilitatea ca două caracteristici să varieze împreună, influențate de un factor comun. Acest fenomen trebuie evitat (a se vedea discuția despre selecția caracteristicilor), dar nu întotdeauna corelația este remarcată. Astfel, o nouă formă x poate să fie mai apropiată de centrul clasei greșite. O soluție posibilă este utilizarea unei distanțe de tip *Mahalanobis*. O problemă similară apare dacă cele două caracteristici nu sunt scalate corespunzător (de exemplu, una este exprimată în centimetri, iar alta în kilometri).
suprafețe de separație dintre clase neliniare	Suprafețele liniare generate de clasificatorii liniari s-ar putea să nu fie corespunzătoare pentru distingerea unor astfel de clase. Soluții posibile ar fi: redefinirea caracteristicilor, aplicarea distanței *Mahalanobis* sau cazul extrem: folosirea *rețelelor neuronale*.
existența subclaselor 	Se observă că avem cele patru subclase separate prin suprafețe liniare, dar nu putem spune același lucru despre cele două clase (este exemplul clasic al imposibilității implementării funcției *XOR* cu ajutorul perceptronului simplu). O soluție posibilă constă în folosirea unei metode specifice învățării nesupravegheate (cum ar fi metoda nucleelor dinamice) pentru împărțirea claselor în subclase. Rezultă astfel patru clase care într-o primă fază pot fi considerate distincte, iar apoi rezultatele sunt combinate cu ajutorul unei funcții *OR*.

Tabelul 1: Limitări ale funcțiilor discriminant liniare

```
repetă
   învățat ← 1
   pentru k ← 0, p-1 execută
   calculează z(k) și y(k) = f(z(k)) pentru forma
   x(k)
   dacă y(k)*z(k)<0 atunci
     modifică valorile ponderilor ω_j, j = 0, ..., n
        învățat ← 0
   sfârșit dacă
  sfârșit pentru
până când învățat = 1
```

Perceptronul multistrat reprezintă o dezvoltare a perceptronului simplu, fiind organizat pe mai multe niveluri, având drept obiectiv realizarea unei funcții de decizie, care transformă vectori aparținând spațiului R^n în vectori m-dimensionali (din R^m). Structura unui perceptron multistrat complet conectat cu trei straturi, n intrări și m ieșiri este prezentată în figura 4 ($\omega_{ij}^{(in)}$ reprezintă ponderea legăturii dintre neuronul i al stratului de intrare și neuronul j al stratului ascuns, iar $\omega_{jk}^{(out)}$ ponderea legăturii dintre neuronul j al stratului ascuns și neuronul k aparținând stratului de ieșire).

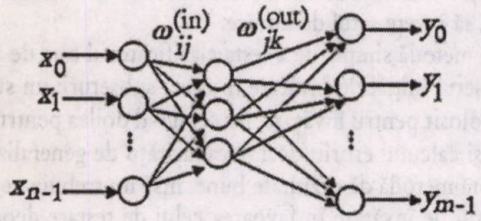


Figura 4

Unul din rezultatele legate de perceptronul multistrat și prin care se explică performanțele deosebite obținute prin utilizarea acestuia este dat de teorema lui *Kolmogorov* care afirmă că orice funcție continuă definită pe un cub n-dimensional (de exemplu $[0, 1]^n$) poate fi scrisă sub următoarea formă:

$$g(x_0, x_1, \ldots x_{n-1}) = \sum_{j=0}^{2n} \Psi_j \left(\sum_{i=0}^{n-1} \phi_{ij}(x_i) \right),$$

unde Ψ_j și ϕ_{ij} sunt funcții continue. Ca urmare, transformarea dintr-un cub n-dimensional $[0, 1]^n$ în spațiul R^m poate fi realizată doar printr-o rețea cu două straturi (fără a număra stratul de intrare) care necesită un strat ascuns cu cel puțin $2 \cdot n + 1$ neuroni. Amintind discuția referitoare la limitările clasificatorului liniar și complexitatea suprafețelor de separare, rezultă imediat aplicabilitatea perceptronului multistrat în discriminarea dintre clase. Trebuie precizat faptul că teorema lui *Kolmogorov* asigură doar existența funcțiilor, nu și forma acestora.

În continuare va fi schițat algoritmul de învățare cu propagarea înapoi a erorii pentru o rețea cu trei straturi. Sunt folosite următoarele notații:

- n - numărul intrărilor rețelei;
- h - numărul neuronilor din stratul ascuns;
- m - numărul ieșirilor rețelei;
- $w_{ij}^{(in)}$ - ponderea legăturii dintre neuronul i din stratul de intrare și neuronul j din stratul ascuns;
- $w_{jk}^{(out)}$ - ponderea legăturii dintre neuronul j al stratului ascuns și neuronul k al stratului de ieșire;
- f - funcția de activare a ieșirii fiecărui neuron; are forma: $f(a) = \frac{1}{1+e^{-a}}$;
- f' - derivata funcției de activare;
- p - numărul de elemente ale setului de învățare;
- x^s - vectorul de intrare (n-dimensional) cu indicele s din setul de învățare: $x^s = (x_0^s, x_1^s, \ldots, x_{n-1}^s)$;
- c^s - vectorul de ieșire așteptat (m-dimensional) când la intrare se aplică forma x^s: $c^s = (c_0^s, c_1^s, \ldots, c_{m-1}^s)$;
- y^s - vectorul de ieșire al rețelei (m-dimensional) când la intrare se aplică forma x^s: $y^s = (y_0^s, y_1^s, \ldots, y_{m-1}^s)$;
- a_j - valoarea de activare a neuronului j din stratul ascuns;
- b_j - ieșirea neuronului j din stratul ascuns $b_j = f(a_j)$;
- o_k - valoarea de activare a neuronului k din stratul de ieșire; evident, y_k^s (ieșirea neuronului k din stratul de ieșire) este $f(o_k)$;
- E_s - eroarea prezentată de ieșirile neuronilor corespunzători intrării x^s;
- E - eroarea medie pentru cele p forme ale intrării;
- *eps* - eroarea acceptată pentru învățarea setului de forme.

Algoritmul de învățare este prezentat în continuare:

Se aleg valori aleatoare mici pentru ponderile rețelei $w_{ij}^{(in)}$ *și* $w_{jk}^{(out)}$, i = 0, ..., n - 1, j = 0, ..., h - 1, k = 0, ..., m - 1

Se alege valoarea constantei de instruire α $(0 < \alpha < 1)$

repetă

pentru s ← 0, p - 1 **execută**

Se calculează ieșirile pentru cele două straturi

$$b_j \leftarrow f(a_j),\ a_j \leftarrow \sum_{i=0}^{n-1} w_{ij}^{(in)} \cdot x_i^s,\ j = 0, \ldots, h-1$$

$$y_k^s = f(o_k),\ o_k = f\left(\sum_{j=0}^{h-1} w_{jk}^{(out)} \cdot b_j \right),\ k = 0, \ldots, m-1$$

Se actualizează ponderile de ieșire $w_{jk}^{(out)}$

$$w_{jk}^{(out)} \leftarrow w_{jk}^{(out)} + \alpha \cdot (c_k^s - y_k^s) f'(o_k^s) f'(a_j)$$

Se actualizează ponderile de intrare $w_{ij}^{(in)}$

$$w_{ij}^{(in)} \leftarrow w_{ij}^{(in)} + \alpha \cdot x_i^s \cdot f'(a_j) \sum_{k=0}^{m-1} w_{jk} \cdot (c_k^s - y_k^s) f'(o_k)$$

Se calculează eroarea rețelei pentru intrarea x^s

$$E_s = \frac{1}{2} \sum_{k=0}^{m-1} (y_k^s - c_k^s)^2$$

sfârșit pentru

Se calculează eroarea medie totală $E = \frac{\sum_{s=0}^{p-1} E_s}{p}$

până când E < eps

Practic, algoritmul constă în minimizarea erorii pentru fiecare formă aplicată la intrare prin actualizarea ponderilor de intrare și de ieșire folosind o metodă de tip gradient (care determină apariția derivatei funcției de activare *f*).

Regula lui Bayes

O altă abordare în cadrul contextului supravegheat al recunoașterii formelor apelează la teoria probabilităților. Fundamentarea deciziei de apartenență a unei forme *x* la o anumită clasă poate fi influențată de următoarele elemente: probabilitatea de apariție a unei anumite forme *x*, probabilitatea de apariție a unei forme *x* aparținând unei clase *j* etc.

Probabilitatea condiționată a evenimentului *A*, știind că un eveniment *B* a avut loc, se definește astfel:

$$\Pr\{A \mid B\} = \frac{\Pr\{A \cap B\}}{\Pr\{B\}},$$

unde evenimentele *A* și *B* sunt considerate submulțimi ale unui spațiu de selecție *S*.

Teorema lui *Bayes* permite calculul probabilităților condiționate astfel:

$$\Pr\{A \mid B\} = \frac{\Pr\{A\} \cdot \Pr\{B \mid A\}}{\Pr\{B\}}.$$

Particularizând pentru problema recunoașterii formelor, fie:

- $\Pr\{x\}$ - probabilitatea de apariție a unei forme particulare *x*;
- $\Pr\{x \mid j\}$ - probabilitatea condiționată de apariție a unei anumite forme *x*, dată fiind clasa *j*;
- $\Pr\{j\}$ - probabilitatea de apariție a unei forme din clasa *j*;
- $\Pr\{j \mid x\}$ - probabilitatea condiționată de apartenență la clasa *j* (probabilitatea de apariție a clasei *j*) dată fiind forma *x*.

Aplicând teorema lui *Bayes* (fiind interesați de probabilitatea de apartenență a unei forme *x* la o clasă *j*), avem:

$$\Pr\{j \mid x\} = \frac{\Pr\{j\} \cdot \Pr\{x \mid j\}}{\Pr\{x\}}.$$

Ca urmare, se poate considera următoarea decizie de clasificare:

$$x \in F_j \Leftrightarrow \Pr\{j \mid x\} = \max\{\Pr\{k \mid x\} / k = 0,\ldots,m-1\}.$$

Așadar, obținem:

$$\Pr\{j \mid x\} > \Pr\{k \mid x\} \Leftrightarrow \Pr\{j\} \cdot \Pr\{x \mid j\} > \Pr\{k\} \cdot \Pr\{x \mid k\}$$

$$\forall k = 0,\ldots,m-1, k \neq j.$$

Se pot defini, de asemenea, funcții discriminant astfel:

$$g_j(x) = \Pr\{j\} \cdot \Pr\{x \mid j\}, j = 0,\ldots,m-1$$

decizia luându-se prin determinarea maximului acestor funcții, cum s-a arătat anterior.

Pentru această abordare sunt necesare două tipuri de informații:

- $\Pr\{j\}$ - care este raportul dintre numărul de forme n_j care aparțin clasei *j* și numărul total de forme din mulțimea de învățare;
- $\Pr\{x \mid j\}$ - care se determină cu ajutorul distribuției normale *Gauss*:

$$\Pr\{x \mid j\} = \frac{1}{\sqrt{2\pi\sigma_j^2}} \cdot e^{\frac{(x-m_j)(x-m_j)^T}{2\sigma_j^2}}$$

unde σ_j reprezintă abaterea medie pătratică iar m_j vectorul formă mediu al clasei *j*.

Algoritmul de învățare și clasificare cuprinde următoarele etape:

- calcularea vectorilor formă medii m_j și a abaterilor pătratice medii σ_j pentru fiecare clasă din setul de învățare ($j = 0, \ldots, m-1$);
- calcularea probabilităților $\Pr\{x \mid j\}$ și $\Pr\{j\}$ pentru fiecare clasă *j*;
- forma nouă *x* este clasificată la clasa *j* dacă:

$$\Pr\{j\} \cdot \Pr\{x \mid j\} = \max\{\Pr\{k\} \cdot \Pr\{x \mid k\} / k = 0,\ldots,m-1\}.$$

Validarea învățării

După implementarea unui model de clasificator în cadrul unui proces de învățare, acesta va trebui să fie capabil de a realiza funcția de generalizare, adică de a lua decizii corecte pentru alte forme decât cele care aparțin setului de învățare. Capacitatea unui clasificator de a generaliza procesul decizional este dată de mărimea erorii la ieșire. Această eroare poate fi realizată pe baza setului de învățare, calculându-se (eventual sub formă de medie pătratică) diferența dintre ieșirile clasificatorului și ieșirile dorite. Însă, eroarea nu ne va da nici o informație despre capacitatea de generalizare a clasificatorului ci doar despre cât de bine a reușit acesta să învețe setul de intrare.

O metodă simplă de a testa clasificatorul este de a împărți setul inițial de învățare în două subseturi: un subset va fi folosit pentru învățare, iar cel de-al doilea pentru testarea și calculul erorii (deci, a capacității de generalizare). Această metodă dă rezultate bune, însă nu trebuie neglijat subsetul de învățare în favoarea celui de testare deoarece aceasta conduce la scăderea performanțelor clasificatorului. O derivare a acestei metode constă în împărțirea setului inițial de învățare în *k* subseturi de mărime egală (de exemplu, $k = 10$) din care numai un subset este folosit pentru testare. Clasificatorul este construit de *k* ori, de fiecare dată folosindu-se un alt subset pentru testare și se păstrează acea formă a clasificatorului care minimizează eroarea la ieșire.

Considerații privind setul de învățare

Din cele prezentate până acum, reiese foarte clar că performanțele clasificatorului (indiferent cum a fost implementat - *NN*, *KNN*, *rețele neuronale* etc.) precum și capacitatea sa de a generaliza sunt influențate semnificativ de setul de învățare folosit.

Un set de învățare ideal cuprinde un număr minim de exemple necesare construirii unui clasificator capabil de a realiza funcția de generalizare în vederea obținerii unui rezultat corect. Deoarece în practică nu se dispune de un asemenea set ideal, se impune realizarea unor operații de "editare" a setului de învățare care poate avea ca scop: eliminarea cazurilor conflictuale (exemple aproape identice, dar a căror clase diferă semnificativ), exemplelor irelevante care, pur și simplu, măresc inutil dimensiunea setului etc. Această operație trebuie realizată astfel încât să nu se modifice

(sau să se modifice nesemnificativ) suprafețele de separație dintre clase deci, fără a afecta calitatea procesului de învățare. De asemenea, timpul necesar învățării (sau clasificării în cazul regulii *NN* și *KNN* unde este necesar calculul a *p* distanțe) se poate reduce considerabil.

În vederea reducerii dimensiunilor setului de învățare prin eliminarea exemplelor redundante, care nu influențează major performanța procesului de învățare, se pot folosi o serie de metode care se bazează pe ***grafurile de proximitate***.

Aceste grafuri de proximitate au ca noduri punctele din spațiul caracteristicilor asociate fiecărui exemplu aparținând setului de învățare. Două puncte (noduri) sunt legate printr-o muchie dacă sunt "apropiate" dintr-un anumit punct de vedere și dacă nu există alte exemple aflate într-o zonă "interzisă" determinată de cele două puncte inițiale. Aceste aspecte sunt clarificate în continuare, considerându-se două exemple.

Diagrama Voronoi

Diagrama Voronoi reprezintă o partiție a spațiului caracteristicilor în regiuni astfel încât toate punctele dintr-o regiune sunt mai apropiate (din punct de vedere al unei anumite distanțe) de un anumit nod decât de celelalte. *Triangularizarea Voronoi* se obține prin unirea prin muchii a nodurilor care fac parte din regiuni ce prezintă o margine comună (figura 5).

Considerând regula *NN*, când o formă necunoscută "cade" într-o anumită regiune, ea va fi clasificată în clasa nodului regiunii respective.

Suprafața de separare dintre cele două clase este alcătuită din segmente reprezentate de margini ale *regiunilor Voronoi*. Ca urmare, nodurile care nu au contribuit la realizarea suprafeței de separare sunt redundante și pot fi eliminate.

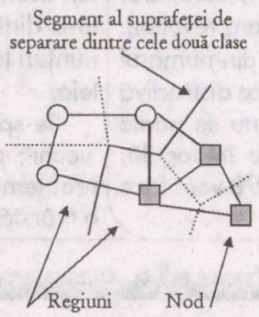


Figura 5

Determinarea triangularizării *Voronoi* se poate realiza folosind un algoritm de tip *divide et impera*, împărțind succesiv setul de învățare în două subseturi de dimensiuni egale, determinând triangularizarea pentru cele două subseturi și realizând reuniunea acestora. În cazurile particulare a două sau trei noduri, regiunile *Voronoi* sunt delimitate de mediatoarele (hiperplanele mediatoare pentru spații cu mai multe dimensiuni) ale segmentelor care unesc cele două, respectiv trei puncte.

Graful Gabriel

În cazul *grafului Gabriel*, două noduri *A* și *B* sunt legate printr-o muchie dacă sfera determinată de diametrul *AB* nu conține alte noduri. În exemplul din figura 6, nodurile *A* și *B* sunt *vecini Gabriel* în timp ce nodurile *A* și *C* nu îndeplinesc această proprietate. Nodurile ale căror vecini din *graful Gabriel* care aparțin aceleiași clase pot fi eliminate deoarece nu contribuie la alcătuirea suprafeței de separare.

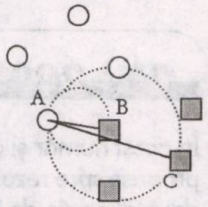


Figura 6

Întotdeauna, setul editat cu ajutorul *grafului Gabriel* este mai mic decât cel obținut cu *diagrama Voronoi* datorită faptului că *graful Gabriel* este un subgraf al *triangularizării Voronoi*.

Algoritmii de editare a setului de învățare sunt identici pentru cele două abordări și constau în următorii pași:

- se determină graful de proximitate (triangularizarea *Voronoi* sau graful *Gabriel*) pentru mulțimea exemplelor din setul de învățare;
- se parcurg nodurile grafului și se marchează acele noduri pentru care toți vecinii aparțin aceleiași clase;
- se șterg nodurile marcate.

Se observă că ordinea de parcurgere a grafului nu contează întrucât toate nodurile marcate sunt șterse în același timp, și anume în cadrul celui de-al treilea pas.

Radu-Daniel Vatavu este student în anul IV la Universitatea Ștefan cel Mare din Suceava. Poate fi contactat prin e-mail la **raduvro@yahoo.com**.

Un nou Athlon XP

Pe data de 1 octombrie 2002 compania Advanced Micro Devices (AMD) a lansat pe piață o versiune îmbunătățită a procesoarelor Athlon XP, destinate sistemelor desktop.

Cea mai importantă noutate introdusă este magistrala de 333 MHz, folosită în premieră în industria procesoarelor.

Au fost lansate două modele ale noului procesor, numite Athlon XP 2700+ și Athlon XP 2800+. Indicativul de performanță sugerează faptul că cel mai rapid procesor AMD este echivalent cu un Intel Pentium 4 la 2,8 GHz, chiar dacă frecvența de funcționare este doar cu puțin mai mare decât 2 GHz.

Din nefericire pentru AMD, noile procesoare sunt disponibile în număr limitat, iar utilizatorii obișnuiți le vor putea achiziționa doar la începutul anului 2003.

Următorul pas important al companiei AMD va fi făcut în cursul anului 2003 când se vor lansa primele procesoare care folosesc o arhitectură pe 64 de biți. O denumire probabilă a viitorului procesor este Athlon DT, iar indicativul de performanță va fi de peste 3000+.

În numărul URMĂTOR

Ultimul număr din acest an al Gazetei de Informatică va apărea la începutul lunii decembrie. Chiar dacă în acel moment veți fi deja pregătiți pentru vacanța de iarnă, veți putea citi, printre altele, articolele prezentate în continuare...

ONI 2002

În acest număr și cel anterior au fost prezentate soluțiile propuse spre rezolvare la ediția din acest an a Olimpiadei Naționale de Informatică. Inițial, am intenționat să publicăm soluțiile autorilor dar, într-un moment de inspirație, am decis că poate ar fi mai bine să existe soluții GInfo ale problemelor de la ONI 2002. Așadar, în timpul acestei veri, ne-am pus în locul concurenților și am început să rezolvăm. Și apoi am comparat soluțiile... și ne-am cam mirat (mai bine zis chiar ne-am revoltat). Dacă vreți să aflați care au fost motivele, citiți articolul din numărul următor. Să nu vă mirați când veți găsi articolul în rubrica distractivă (Enjoy)! Ne-am dat seama că nu avem ce face, că nu se poate schimba nimic fiindcă, oficial, întotdeauna totul este în regulă; așadar, conform bunei tradiții a românilor, în GInfo 12/8 vom face "haz de necaz".

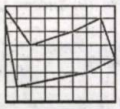
Amintiri

În acest număr ne-am amintit de problemele pe care le rezolvau elevii clasei a IX-a la olimpiadele județene din 1994.

Pentru numărul următor vom deschide Gazeta de Informatică nr. 5 din același an și vom transforma câteva dintre problemele ale căror enunțuri le vom găsi în paginile acesteia.

Se spune că nu există probleme "vechi"; pentru un nou-născut orice problemă este nouă, chiar dacă are o "vârstă" de aproape 10 ani.

Circuite bitonice



Dacă, după ce ați citit articolul despre ONI 2002 veți considera că mai are rost să vă pregătiți pentru concursurile de informatică (noi sperăm că am exagerat și că nu are rost să vă descurajați), veți putea citi o prezentare a modului în care poate fi rezolvată o problemă care, pe parcursul anilor, a devenit celebră și anume problema determinării celui mai scurt circuit bitonic corespunzător unei mulțimi de puncte din plan.

Bursele Agora

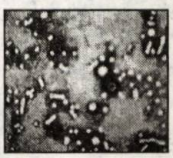
În numărul următor vom prezenta soluțiile problemelor de la primele patru etape, clasamentul intermediar, precum și enunțurile problemelor de la etapele care s-au desfășurat între timp.

ACM 2002

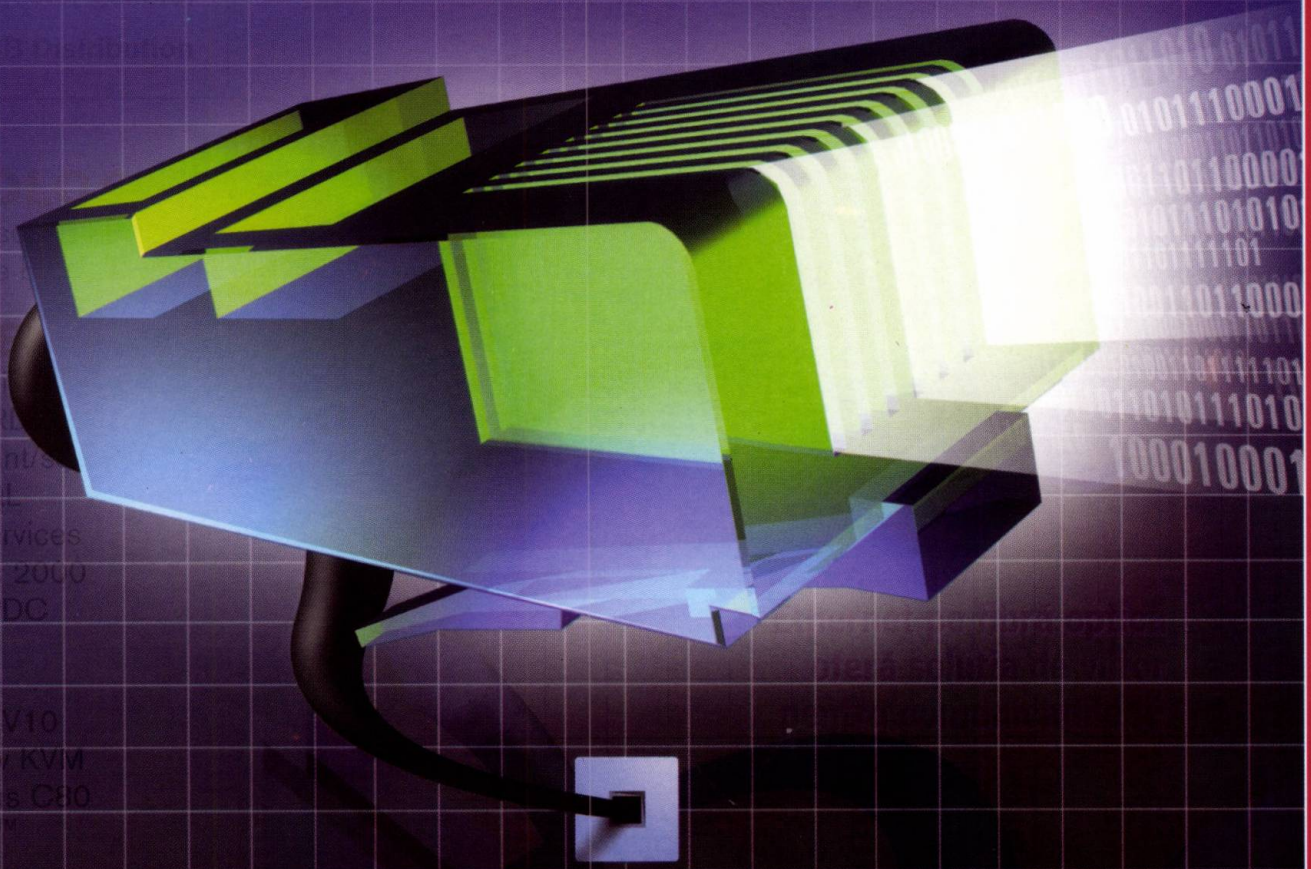
Etapa regională pentru Europa de Sud-Est a concursului ACM se va desfășura la București pe data de 19 octombrie. În numărul următor veți găsi enunțurile problemelor propuse spre rezolvare la această competiție.

Recunoașterea formelor

În numărul următor veți putea citi ultimul episod al serialului care prezintă tehnici folosite pentru a recunoaște sau clasifica diferite forme. Vor fi prezentate mai multe procedee de învățare nesupravegheată cum ar fi metoda celor mai mici pătrate, metoda nucleelor dinamice sau metode ierarhice de clasificare. Serialul se va încheia cu o prezentare comparativă a clasificatorilor descriși pe parcursul celor trei episoade.



NETREPORT



Conectat la viitorul digital

AGORA MEDIA: Str. Tudor Vladimirescu nr. 63/9, Tg. Mureș, CP 230/1, 4300
Tel: 065-166 516, Fax: 065-166 290, e-mail: office@agora.ro

www.netreport.ro

Agora MEDIA

Publicarea acestei reviste este posibilă datorită sponsorizărilor acordate de următoarele firme care s-au decis să sprijine învățământul românesc de informatică

SPONSORI DE PLATINĂ

COMPAQ

SPONSORI DE AUR

FORTE COMPUTERS
Flexibility. Made in Romania

ASTRAL TELECOM

SPONSORI DE ARGINT

sdt
IT SOLUTIONS & SERVICES

hp
invent

ERICSSON

Magua

M.G.T.
EDUCATIONAL

a&C INTERNATIONAL S.A.
the CAD-CAMpany
autodesk
authorized distributor

ALCATEL

*Rugăm companiile care doresc să se alăture acestui efort să ne contacteze prin e-mail la adresa office@ginfo.ro sau la numărul de telefon: 065-166.516

Decembrie 2002
Vol. 12/8

Lei 30.000

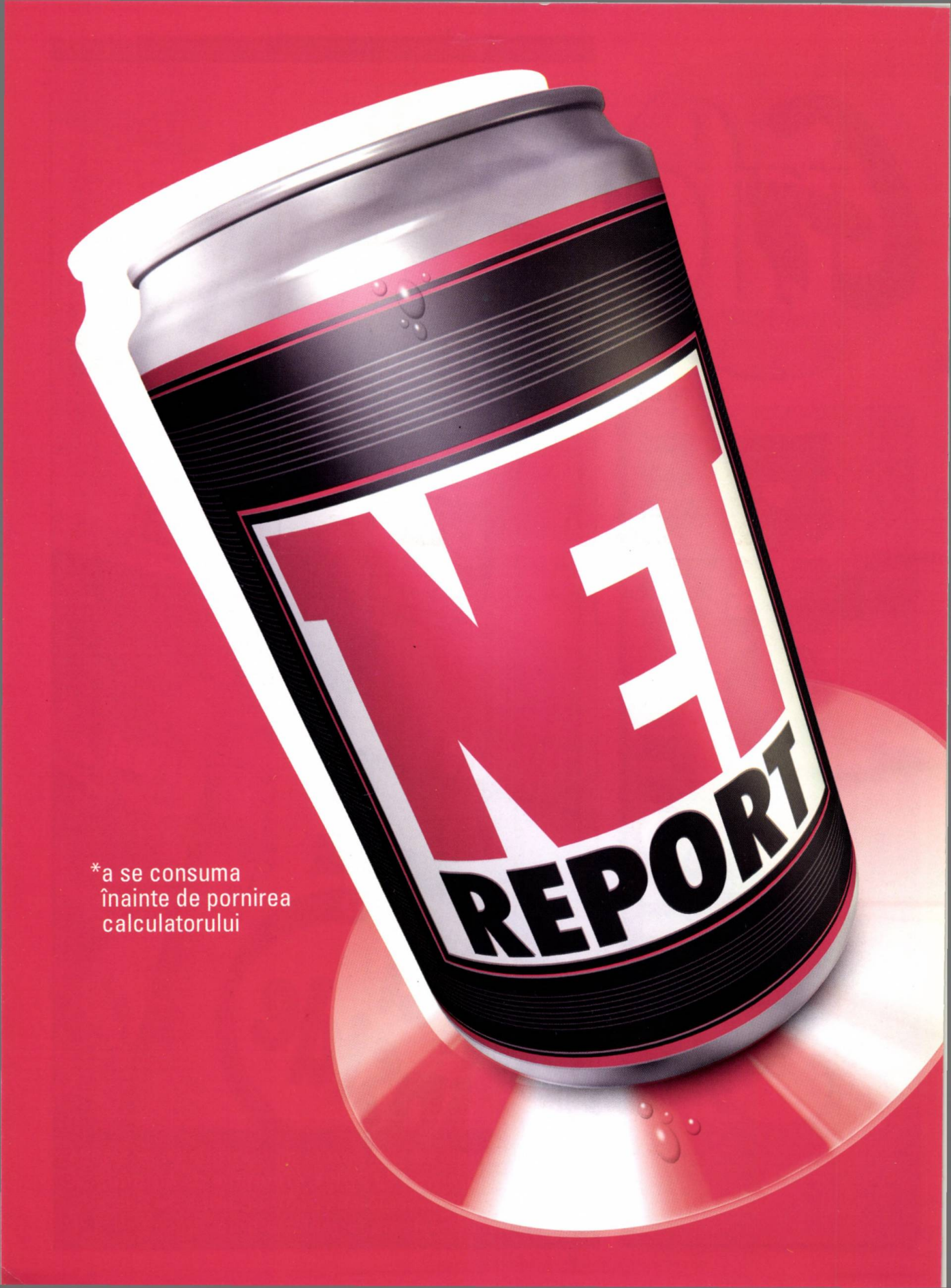
Gazeta de informatică

Revista viitorilor specialiști în tehnologia informației

din sumarul numărului:

Plus

Probleme preluate din Gazeta de Informatică nr.5/1994
Internet-pagini web cu PHP 4 • Teoreme de programare
ACM 2002-etapa regională pentru Europa de Sud-Est
Recunoașterea formelor-ultimul episod



*a se consuma
înainte de pornirea
calculatorului

Vol. 12, Nr. 8 /2002

gazeta de informatică

Redacția:

Director științific:
prof. dr. *Horia Georgescu*
Director de publicație:
lect. univ. *Clara Ionescu*
Redactor-șef:
Mihai Scorțaru
Redactor-șef adjunct:
Carmen Șușman
Redactori:
prof. *Eugen Ionescu*
Claudiu Soroiu
Au mai colaborat:
prof. *Lucian Ilea (Cluj-Napoca)*
Vlad Țira (Cluj-Napoca)
Radu-Daniel Vatavu (Suceava)
Tehnoredactare:
Claudiu Soroiu
Coperta și grafica:
Andrei Pașa

Adresa pentru corespondență:
Computer Libris Agora,
Universității 8/8,
3400, Cluj-Napoca
Adresa e-mail: office@ginfo.ro
Tel: 0264/192.422
0265/166.516
Fax: 0265/166.290
http://www.ginfo.ro
ISSN: 1222-7129

Tiparul: *Master Druck* Târgu Mureș
tel: 0265/162.359
Editor: Agora Media SRL
Agora Media SRL, editor al publicațiilor "*eWeek România*", "*PC Magazine România*", "*NetReport*" și "*Gazeta de Informatică*", este membru al Biroului Român pentru Auditarea Tirajelor.
str. T. Vladimirescu nr. 63/9
4300 Târgu-Mureș
Director general:
Romulus Maier
Director executiv:
Adrian Pop
Director tehnic:
Szabó László
Vânzare reclamă:
Mihai Bucuroiu
Cosmin Ursu
Denis Marioara
Marius Comărniceanu

Reprezentanța București:
Director: *Dana Pastramă*
Tel: 021/330.92.81
Fax: 021/330.92.85

Sumar

actual.. PAG. 5

Dincolo de 3 GHZ
Mihai SCORȚARU

probleme.. PAG. 7

ACM 2002

~ etapa regională pentru ~
~ Europa de Sud-Est ~

AMINTIRI

~ probleme din Gazeta ~
~ de Informatică 5 / 1994 ~

soluții.. PAG. 14

Olimpiada INTERNAȚIONALĂ de Informatică

~ Yong-In, Coreea de Sud ~

concurs.. PAG. 21

Despre primele patru runde

Clasament

Enunțurile problemelor de la rundele #05-#08

focus.. PAG. 28

Circuite BITONICE
Lucian ILEA
Vlad ȚIRA

internet.. PAG. 30

Pagini WEB cu PHP 4
Mihai SCORȚARU
Claudiu SOROIU

serial.. PAG. 32

RECUNOAȘTEREA formelor
Radu-Daniel VATAVU

~ elemente de grafică ~
~ ultimul episod ~
~ al serialului ~

begin.. PAG. 37

TEOREME de programare
Clara IONESCU

~ teorie și practică ~

enjoy.. PAG. 40

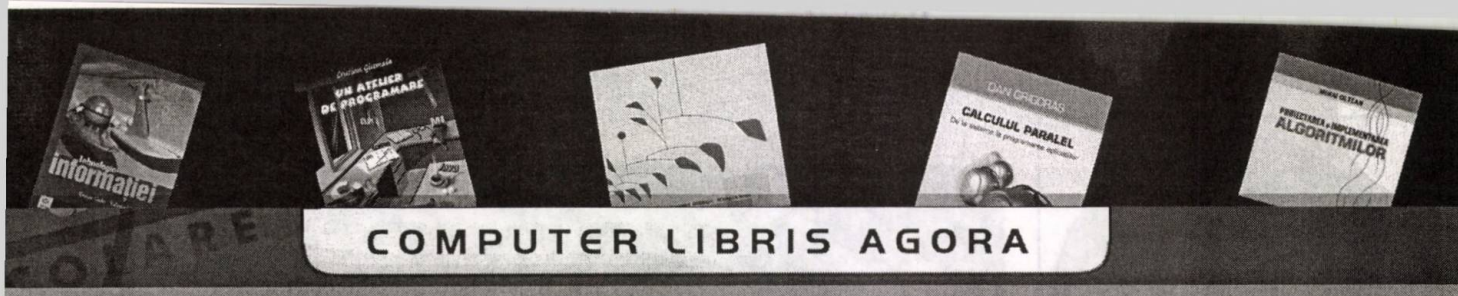
După ONI 2002
Mihai SCORȚARU

~ studiu de (ne)caz ~

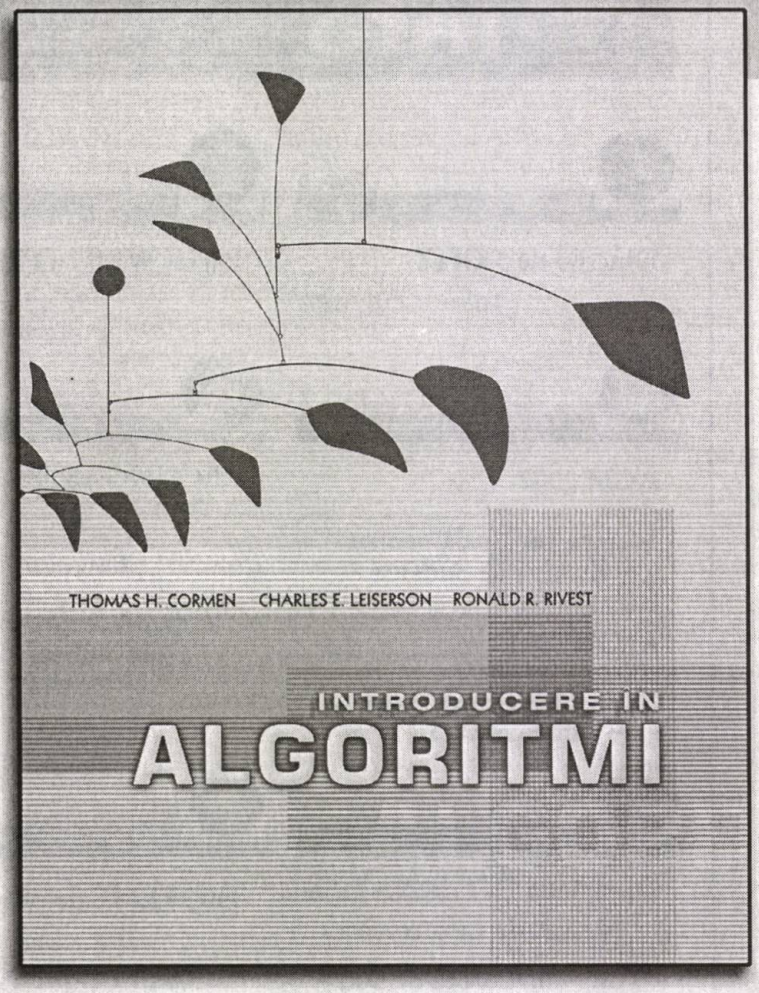
dialog.. PAG. 44

POȘTA redacției

~ așteptăm scrisorile voastre ~



COMPUTER LIBRIS AGORA



Cartea se adresează, în primul rând, celor care studiază la nivel elementar sau academic algoritmi și structuri de date. Deoarece tratează aspecte teoretice și practice, legate de proiectarea algoritmilor, este la fel de binevenită elevilor, studenților, profesorilor și autodidacților.
Introducere în algoritmi acoperă atât metodele clasice cât și metodele moderne ca analiza amortizată și algoritmii paraleli.
Prezentarea matematică, riguroasă, este atent detaliată în așa fel încât să fie accesibilă tuturor cititorilor.
Capitolele încep cu elemente de bază, avansând spre subiecte mai complexe.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

Despre JOCURI și alți DEMONI

Titlul editorialului este inspirat din cel al romanului ***Despre dragoste și alți demoni***, scris de celebrul autor columbian, *Gabriel Garcia Marquez*. Dacă nu ați citit cartea, v-o recomand. Mai trebuie să scăpați și voi puțin de programare... Am auzit că e o carte excelentă, deși nu am avut timp să o citesc până acum.

Revenind la titlul editorialului (și la "fuga" de programare) nimeni nu poate spune că există persoane care nu se mai joacă puțin dacă au acces la un calculator. Mai sunt și excepții, dar cam puține... Așadar, cel puțin un ***FreeCell*** sau un ***Solitaire*** se cam joacă toată lumea. Este perfect normal să beneficiezi de cât mai multe dintre facilitățile oferite de calculator. Nu voi mai repeta ceea ce s-a tot scris în revistele de specialitate (și nu numai). Știți deja povestea... e periculos dacă se exagerează.

Problema e cu pauzele... care ar trebui să fie de programare. Din nefericire, acum când liceele au fost dotate cu calculatoare performante, se face programare în pauzele de jocuri. În recreații pasionații de "informatică" discută despre noi strategii pentru *Warcraft III*, tactici pentru *Counter-Strike*, se laudă cu obiectele magice pe care le-au găsit prin *Dungeon Siege*, *Neverwinter Nights*, *Morrowind* etc. Când eram eu în liceu se mai discuta și despre cum se desenează obiecte tridimensionale sau cum se găsesc numele tuturor fișierelor dintr-un director folosind funcțiile *unit*-ului `Dos` din *Borland Pascal*. Sunt vremuri de mult apuse... Aveam și noi jocuri interesante cum ar fi *Scorch*, *Bomber Man* etc., dar parcă ne jucam mai rar. Unii dintre noi creau jocuri noi și apăreau competiții între colegi. Îmi aduc aminte când am creat un "joc" care avea ca scop dezvoltarea dexterității în scrierea textelor. Timp de câteva săptămâni pe majoritatea monitoarelor calculatoarelor din laboratoare apărea jocul meu. Să vedeți cum se lăuda câte unul că a intrat în primii zece! Mai participau și profesori... Era parcă altceva... Acum elevii să laudă că au cel mai mare *Strength*, că i-au depășit la *frag*-uri pe ceilalți etc.

Trebuie să recunosc că mă joc destul de des. Poate mai mult decât ar trebui. Poate mai exagerez câte o dată... mai ales cu jocurile din seria *Heroes of Might and Magic*.

Recent am primit jocul *Medal of Honour: Allied Assault*. Nu vă impacientați că nu l-am cumpărat. L-am câștigat și eu la un fel de tombolă. Încă nu îmi pot permite să plătesc aproape un milion de lei pe un joc (care, pe deasupra, nici nu-mi place).

Ceea ce m-a impresionat cel mai mult a fost prima pagină a manualului de utilizare. Aici erau "instrucțiunile de protecția muncii". Prima dintre ele (scrisă cu litere de o șchioapă) continea un avertisment. Se spunea că jocul poate provoca crize de epilepsie chiar dacă *gamer*-ul nu a mai suferit astfel de crize înainte. Este evident faptul că avertismentul este scris doar pentru a evita viitoarele procese care ar avea loc dacă vreun jucător ar suferi astfel de crize, dar parcă îți cam taie pofta de joc.

Se mai spunea că nu este recomandat să se folosească ecrane mari în timpul jocului, că încăperea trebuie să fie perfect luminată, că după fiecare 45 de minute de joc trebuie făcută o pauză de cel puțin un sfert de oră etc. Cam multe avertismente, nu credeți?

Voi aborda în continuare un subiect care nu convine deloc *gamer*-ilor. ***Jocurile costă!*** Din nefericire, liceele românești încurajează pirateria prin faptul că este permisă instalarea jocurilor obținute ilegal. Se spune că este inadmisibil ca un joc să coste cât un sfert din salarul mediu pe economie și așa este! Totuși, puțini spun că nu prea e frumos ***să furi***.

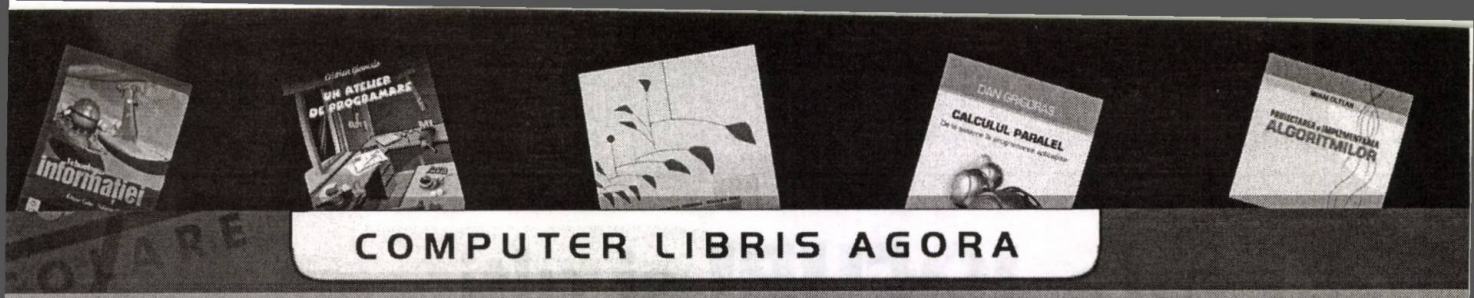
Dacă s-ar face un clasament cu numărul de jocuri deținute legal de români cred că aș fi printre primii cu cele 20 ale mele. E adevărat că doar pe unul l-am cumpărat (*Heroes III*), celelalte fiind cadouri primite cu diferite ocazii. Totuși, este inadmisibil să fie încurajată pirateria. E oarecum normal să o tolerăm (și marii producători de *software* o fac), dar nu să ajutăm la dezvoltarea ei.

Cam atât despre jocuri (de data aceasta)... Voi trece acum la ceilalți "demoni". Cel mai înspăimântător dintre ei mi se pare *IRC*-ul. Unii elevi "se dau pe *mIRC*" de dimineața până seara. Ce discută ei pe-acolo nu prea îmi pot imagina, dar efectele se văd destul de repede. Dificultatea de integrare în societatea "reală" este doar unul dintre ele. Mai sunt și altele, la fel de grave, cum ar fi reducerea capacității de concentrare, probleme de vedere etc.

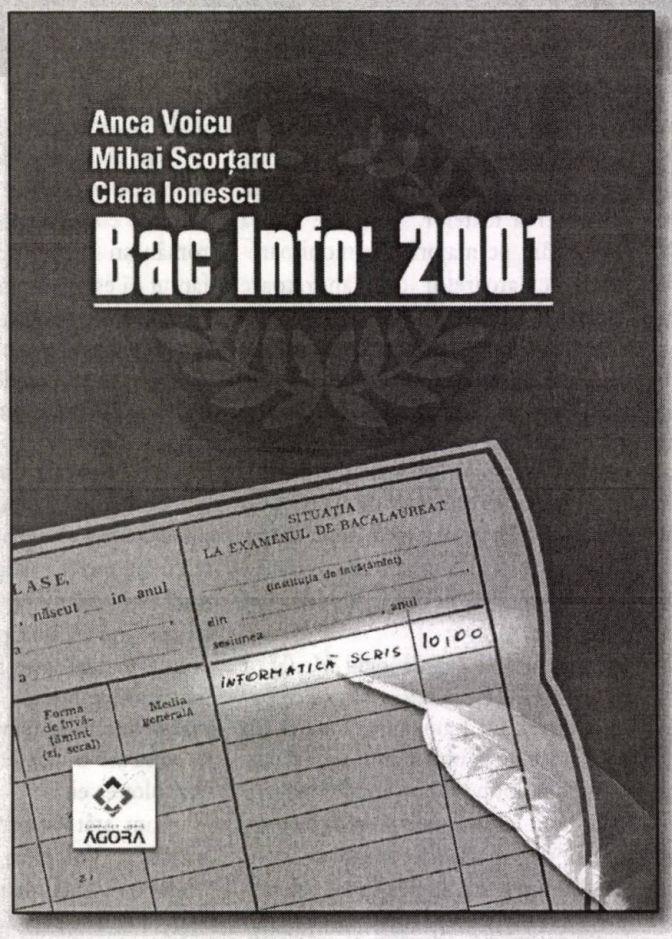
În fine, mai sunt și alți "demoni"; au nume interesante: *WWW*, *Video-CD*, *DVD* etc.

Nu pot să închei decât urându-vă *Distracție plăcută în fața monitorului* și exprimându-mi speranța că nu veți exagera.

Mihai Scorțaru



COMPUTER LIBRIS AGORA



Bac Info' 2001 conține 20 de variante de subiecte pentru Bacalaureat. Fiecare variantă conține câte patru subiecte. Primul dintre ele conține trei subpuncte care testează capacitățile candidaților de a recunoaște valoarea de adevăr a unei propoziții, de a determina efectul unui anumit algoritm și de a găsi o anumită greșeală într-un algoritm. Al doilea subiect conține doua subpuncte; primul testează deprinderea candidaților de a folosi limbajul pseudocod, iar al doilea cea de a folosi un limbaj de programare (Pascal sauC++). Al treilea subiect este o problemă pentru rezolvarea căreia este necesară folosirea unor structuri dinamice de date, iar al patrulea este o problemă pentru a cărei rezolvare trebuie folosită metoda backtracking.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

Dincolo de 3 GHZ

Mihai Scorțaru

În domeniul procesoarelor, recordurile de frecvență sunt depășite destul de des. Atingerea anumitor valori ale frecvenței unui nou procesor nu reprezintă, de obicei, decât îmbunătățiri ale versiunilor anterioare. Este mult mai important când, în momentul lansării unui nou procesor, există noutăți tehnologice care duc la creșterea performanțelor nu numai datorită frecvențelor superioare. Pe data de 14 noiembrie compania Intel a lansat primul procesor cu frecvența mai mare decât 3 GHz și, mult mai important, a fost implementată tehnologia Hyper-Threading pentru procesoarele destinate sistemelor desktop.

Dincolo de 3 GHz

Recordurile sunt stabilite pentru a fi depășite. Din acest punct de vedere atingerea barierei de 3 GHz nu reprezintă un eveniment ieșit din comun, deși *Intel* a sărbătorit așa cum se cuvine evenimentul la ediția din această toamnă a celei mai importante expoziții de tehnică de calcul din lume: *COMDEX*.

Cifrele rotunde reprezintă "bariere psihologice" cărora, de obicei, li se dă o mult mai mare importanță decât ar merita. De exemplu, lansarea procesorului *Pentium* 4 cu frecvența de 2 GHz a fost privită ca un eveniment foarte important deși, practic, produsul nu aducea nimic nou cu excepția frecvenței de funcționare.

Situația a fost diferită pe data de 14 noiembrie când *Intel* a anunțat disponibilitatea procesorului *Pentium* 4 cu frecvența de 3,06 GHz. Pe lângă faptul că s-a depășit valoarea de 3 GHz, noul procesor aduce o noutate importantă: tehnologia *hyper-threading* care poate duce la obținerea unor performanțe mai mari comparativ cu un procesor aproape identic, dar care nu beneficiază de acest avantaj.

Deși procesoarele *Intel Xeon* beneficiau de avantajele acestei tehnologii de câteva luni, *Pentium* 4 cu frecvența de 3,06 GHz este primul procesor destinat sistemelor *desktop* care oferă suport pentru această tehnologie. Există zvonuri potrivit cărora suportul pentru *hyper-threading* era prezent în procesoarele *Pentium* 4 încă din momentul în care a fost atinsă frecvența de 2,4 GHz, dar era dezactivat.

Războiul gigahertzilor

Lupta pentru atingerea frecvenței de 1 GHz a fost câștigată pe data de 6 martie 2000 de *Advanced Micro Devices* care a lansat atunci un procesor *Athlon* care funcționa la această frecvență. Replica celor de la *Intel* s-a lăsat așteptată doar două zile, pe data de 8 martie 2000 fiind anunțat un procesor *Pentium III* care funcționa la aceeași frecvență. Din nefericire pentru *Intel*, lansarea a fost doar pe hârtie, procesoarele *Pentium III* la 1 GHz fiind disponibile pentru publicul larg doar pe data de 14 august 2000.

AMD a înregistrat un succes deosebit datorită acestei victorii, supremația companiei *Intel* fiind serios amenințată datorită imposibilității de a oferi procesoare cu viteze mari de funcționare, în timp ce *AMD* nu avea nici o problemă în furnizarea de procesoare cu frecvențe de 1 GHz și chiar mai mult.

Intel și-a dat seama că frecvența de funcționare a procesoarelor are un impact psihologic foarte mare asupra cumpărătorilor și a început să ofere procesoare *Pentium* 4 cu frecvențe mari, chiar dacă performanțele erau inferioare celor ale unor procesoare *AMD* cu frecvențe mai mici de funcționare.

Datorită acestui fapt *Intel*, a fost prima companie care a reușit să atingă, pe data de 27 august 2001, bariera de 2 GHz.

AMD furniza procesoare mai rapide, dar care aveau o frecvență de funcționare mai mică, motiv pentru care cumpărătorii erau atrași de produsele firmei rivale. Din acest motiv, în momentul lansării procesorului *Athlon XP*, a fost introdus un sistem de *rating* al procesoarelor din această familie. De exemplu, un *Athlon XP* cu frecvența de 1,4 GHz era denumit *Athlon XP* 1600+, ceea ce sugera faptul că performanțele erau superioare unui *Pentium* 4 cu frecvența de 1,6 GHz.

Din nefericire, *AMD* nu a reușit să țină pasul cu *Intel* chiar dacă folosea astfel de indicative pentru procesoarele sale. În scurt timp a devenit evident faptul că produsele "de vârf" oferite de *AMD* aveau performanțe

mai mici decât cele mai rapide procesoare *Intel*.

La câteva luni după atingerea barierei de 2 GHz de către *Intel*, *AMD* a reușit să lanseze procesorul *Athlon XP* 2000+, dar care funcționa de fapt la frecvența de 1,66 GHz.

Pe măsura trecerii timpului situația s-a înrăutățit, *AMD* reușind să atingă frecvența de 2 GHz doar pe data de 21 august 2002 (la aproape un an după *Intel*) în momentul lansării procesoarelor *Athlon XP* 2400+.

La data lansării procesorului *Pentium* 4 cu frecvența de 3,06 GHz, cel mai rapid procesor *AMD* era *Athlon XP* 2800+ care funcționa la aproximativ 2,25 GHz.

Despre Hyper-Threading

În principiu, tehnologia *hyper-threading* constă în capacitatea procesorului de a executa simultan două fire de execuție.

Până acum, în momentul execuției unui fir, anumite "zone" ale procesorului erau neutilizate. Acum, aceste "zone" pot executa instrucțiuni ale unui alt fir. Practic, în astfel de situații, calculatorul funcționează ca și cum ar avea două procesoare.

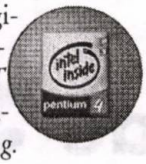
Pentru a putea beneficia de avantajele acestei tehnologii, sistemul de operare și produsele software trebuie să fie scrise astfel încât să profite de această capacitate a procesorului. Singurul sistem de operare *Windows* cu suport pentru *hyper-threading* este *Microsoft Windows XP*.

Dacă nu există și suport *software* pentru *hyper-threading*, atunci calculatorul va beneficia doar de avantajele frecvenței mai mari de funcționare a noului procesor.

Intel susține că performanțele unui procesor care suportă *hyper-threading* cresc cu aproape 70% dacă sunt rulate două programe simultan. Cu toate acestea, testele efectuate au demonstrat creșteri de performanță de aproximativ 35%. În unele cazuri, s-au constatat chiar scăderi ale performanței cu câteva procente atunci când calculatorul executa un singur fir de execuție, deci noile facilități erau practic inutile.

Un nou logo

Pentru a sublinia importanța noii tehnologii, *Intel* a modificat sigla procesoarelor *Pentium* 4. După cum se poate observa în imaginea alăturată, pe noul logo apar literele *H* și *T* care indică prezența tehnologiei *hyper-threading*.



Preț și disponibilitate

Ca orice produs "de vârf", noul *Pentium* 4 la 3,06 GHz cu *hyper-threading* are un preț foarte ridicat, ceea ce îl face inaccesibil pentru marea majoritate a utilizatorilor obișnuiți.

Imediat după lansare, procesorul putea fi achiziționat de la *Intel*, în cantități de cel puțin 1000 de bucăți, la prețul de 637$. Din nefericire, procesorul era disponibil în cantități limitate, motiv pentru care prețul din magazine era de peste 900$. Cantități suficiente pentru a acoperi cererile vor exista doar la mijlocul lunii decembrie. O scădere a prețului oficial este preconizată, cel mai devreme, la sfârșitul lunii februarie a anului 2003.

Planuri de viitor

Datorită faptului că prețul noului procesor este destul de mare, se presupune că *Intel* va lansa (mai mult sau mai puțin oficial) și procesoare cu frecvența de 2,8 GHz care vor beneficia de avantajele oferite de tehnologia *hyper-threading*. Evident, prețurile unor astfel de procesoare ar fi mai accesibile publicului larg.

Următoarea noutate pe care o va oferi *Intel* este un procesor fabricat pe baza unei tehnologii de 0,09 μm (actualele procesoare folosesc o tehnologie de 0,13 μm). Acest procesor este cunoscut sub numele de cod *Prescott*, iar lansarea sa este așteptată în primul trimestru al anului 2003. O altă îmbunătățire ar fi trebuit să fie creșterea frecvenței magistralei de la 533 la 667 MHz.

Datorită faptului că *AMD* nu mai reprezintă o amenințare imediată, *Intel* și-a permis să amâne cu un trimestru lansarea acestui procesor. Această amânare va permite creșterea frecvenței magistralei până la 800 MHz, dublarea memoriei *cache* și implementarea tehnologiei *Hyper-Threading II*. Așadar, *Prescott* va apărea în al doilea trimestru al anului 2003, va avea o memorie *cache* de 1 MB (față de cei 512 KB ai actualelor procesoare *Pentium* 4) și va beneficia de avantajele unei tehnologii *Hyper-Threading* îmbunătățite.

Deși s-a anunțat că ultimul procesor *Pentium* 4 fabricat pe baza tehnologiei de 0,13 μm va avea frecvența de 3,06 GHz, în vara acestui an s-a anunțat că va exista și un *Pentium* 4 cu frecvența de 3,2 GHz care va fi lansat la începutul anului următor. Mai mult, se pare că va exista și o versiune cu frecvența de 3,4 GHz care va apărea în al doilea trimestru al anului 2003.

Chiar dacă replica *AMD* se lasă așteptată, această compania va lansa și ea noi produse. Recent a fost anunțată denumirea viitorului procesor pentru *desktop*-uri care va fi construit pe baza unei arhitecturi pe 64 de biți. Denumirea nu este *Athlon DT* așa cum a fost anunțat anterior, ci *Athlon* 64. Aceste procesoare sunt încă așteptate pentru sfârșitul acestui an, dar este foarte probabil ca ele să apară la începutul anului următor. Indicativul de performanță al primului procesor *Athlon* 64 va fi, probabil, 3400+, dar ar putea exista modificări. Valoarea exactă a frecvenței reale de funcționare este încă necunoscută.

Concluzii

Cu această lansare, *Intel* își consolidează poziția de lider pe piața microprocesoarelor. Lansarea de la mijlocul lunii noiembrie reprezintă depășirea unei noi bariere psihologice (3 GHz) pentru frecvența de funcționare a procesoarelor.

Chiar dacă, de obicei, asemenea cifre rotunde nu indică modificări tehnologice importante, prin introducerea *hyper-threading*-ului *Intel* a creat o primă excepție la această "regulă".

Noile procesoare sunt mai rapide pe de o parte, datorită frecvenței mai mari de funcționare și pe de altă parte datorită noii tehnologii care permite folosirea mai eficientă a resurselor procesorului.

ACM 2002

Pe data de 19 octombrie a avut loc la Universitatea Politehnica din București etapa regională pentru Europa de Sud-Est a concursului internațional de programare ACM. Vă prezentăm enunțurile celor opt probleme propuse spre rezolvare la acest concurs.

P080201: Bancomat

O bancă plănuiește să instaleze un nou automat bancar pentru retragere de numerar. Automatul este capabil să furnizeze bancnote ♣ (acest simbol reprezintă moneda folosită în țara în care este instalat bancomatul - poate reprezenta *dolari*, *euro*, *lire sterline* etc.) cu valoarea totală egală cu suma cerută de client. Bancomatul folosește exact N tipuri de bancnote cu valori diferite. Vom nota valorile celor N tipuri de bancnote prin D_k, $k = 1, ..., N$. De asemenea, vom nota prin n_k rezerva (numărul) de bancnote cu valoarea D_k. De exemplu, am putea avea:

$N = 3, n_1 = 10, D_1 = 100, n_2 = 4, D_2 = 50, n_3 = 5, D_3 = 10,$

ceea ce înseamnă că bancomatul are o rezervă de 10 bancnote cu valoarea de 100♣, 4 bancnote cu valoarea de 50♣ și 5 bancnote cu valoarea de 10♣.

Vom nota prin ***cash*** suma cerută de client. Va trebui să scrieți un program care determină cea mai mare sumă posibilă, mai mică sau egală cu ***cash***, care poate fi retrasă, ținând cont de bancnotele din rezerva bancomatului.

Date de intrare

Datele de intrare vor fi citite din fișierul text **A.DAT**. Fiecare linie a acestui fișier reprezintă un set de date (corespunzător unei tranzacții) și are formatul:

cash N n_1 D_1 n_2 D_2 ... n_N D_N,

unde $0 \le cash \le 100000$ reprezintă suma cerută de client, $0 \le N \le 10$ reprezintă numărul tipurilor de bancnote, $0 \le n_k \le 1000$ reprezintă numărul bancnotelor disponibile care au valoarea D_k ($1 \le D_k \le 1000$, k=1, ..., N). Numerele de pe o linie pot fi separate prin oricâte spații, iar datele de intrare sunt întotdeauna corecte.

Date de ieșire

Pentru fiecare set de date, rezultatul va fi afișat la ieșirea standard, fiecare răspuns fiind scris pe câte o linie.

Exemplu

A.DAT	*Ieșire*
735 3 4 125 6 5 3 350	735
633 4 500 30 6 100 1 5 0 1	630
735 0	0
0 3 10 100 10 50 10 10	0

Explicație

Primul set de date reprezintă o tranzacție în care suma cerută este 735♣. Bancomatul are o rezervă de 3 bancnote cu valoarea de 125♣, 6 bancnote cu valoarea de 5♣ și 3 bancnote cu valoarea de 350♣. În aceste condiții bancomatul va putea furniza suma exactă.

Pentru al doilea set de date bancomatul nu va putea furniza suma exactă, iar clientul va putea retrage suma de 630♣. Se observă că există mai multe posibilități de a obține această sumă folosind rezervele bancomatului.

Pentru al treilea set de date bancomatul nu are la dispoziție nici o bancnotă, deci nu este posibilă nici o extragere de numerar, iar pentru cel de-al patrulea set de date suma cerută este 0♣, deci bancomatul nu trebuie să folosească nici o bancnotă pentru a "elibera" această sumă.

P080202: Ținutul berii

Regele *ACM* 2002 a moștenit un teren și o sumă de bani. În testament se specifica clar că moștenitorul trebuie să construiască n orașe. De asemenea, trebuie să construiască străzi (acestea au două sensuri de circulație) între orașe, astfel încât un călător poate ajunge din fiecare oraș în oricare altul chiar dacă una dintre străzi este închisă pentru reparații. Toate străzile trebuie să fie construite în așa fel încât să poată fi privite ca fiind segmente de dreaptă. Mai mult, condiția ca un călător să poată ajunge dintr-un oraș în altul (chiar dacă o stradă este închisă), trebuie să fie satisfăcută în așa fel încât străzile să fie traversate integral. Nu există posibilitatea ca un călător să pornească pe o stradă, să se oprească la o intersecție aflată în afara orașelor și să continue pe o altă stradă.

Viitorii locuitori au decis să producă bere deoarece solul din această regiune permitea plantarea culturilor necesare. Regele a decis ca la fiecare intersecție de străzi (punctul de întâlnire a două sau mai multe străzi) să fie construit un stand cu bere. Așadar, dacă trei sau patru (sau zece) străzi se intersectează într-un singur punct, atunci există un singur loc în care poate fi amplasat standul corespunzător intersecției. Regele trebuie să aleagă pozițiile orașelor și străzile care vor uni orașe. Va trebui să îl sfătuiți cu privire la cea mai bună alegere, astfel încât să poată fi construite cât mai multe standuri cu bere.

Date de intrare

Prima linie a fișierului de intrare **B.DAT** conține numărul seturilor de date. Fiecărui set de date îi corespunde o linie care conține numărul n ($1 \le n \le 32767$) al orașelor care trebuie construite.

Date de ieșire

Pentru fiecare set de date va trebui să scrieți la ieșirea standard câte o linie care va conține numărul maxim de standuri cu bere care pot fi construite.

Exemplu

B.DAT	*Ieșire*
3	0
3	1
4	5
5	

P080203: Iaht

Sunteți fericitul proprietar al unui iaht foarte frumos. Aveți multe cereri de închiriere a iahtului pentru perioada verii și ați hotărât să obțineți un profit cât mai mare. Pentru fiecare client cunoașteți numărul de zile pentru care dorește să închirieze iahtul și o listă de opțiuni. O opțiune constă dintr-o sumă de bani și un termen limită (numărul maxim de zile dintre începutul verii și ziua în care s-ar termina perioada de închiriere pentru clientul respectiv). Veți primi suma corespunzătoare unui termen limită numai dacă reușiți să închiriați iahtul astfel încât călătoria să se încheie cel târziu în ziua corespunzătoare termenului limită.

De exemplu, prietenul vostru *Jack* vrea să facă o călătorie de 20 de zile prin *Marea Mediterană*. Opțiunile pe care vi le pune la dispoziție sunt:

- Dacă termenul limită este de 60 de zile, primiți 1000 de *EURO*; veți primi această sumă numai dacă durata de închiriere durează exact 20 de zile și începe cel târziu în a 41-a zi (60 - 20 + 1) de la începutul verii;
- Dacă termenul este de 70 de zile primiți 800 de *EURO*.

Dacă un client nu contribuie la maximizarea profitului, puteți renunța la clientul respectiv. Îi cunoașteți pe toți clienții și știți care sunt opțiunile fiecăruia. Va trebui să închiriați iahtul în ordinea în care sosesc cererile clienților.

Scrieți un program care, dându-se opțiunile clienților, determină profitul maxim care poate fi obținut.

Date de intrare

Datele de intrare vor fi citite din fișierul text **C.DAT**, fiecare set de date din fișier având următorul format:

- pe prima linie valoarea n - numărul clienților ($n \le 100$);
- pe n linii distincte numărul de zile (cel mult 100) pentru care un client dorește să închirieze iahtul;
- pe următoarea linie numărul total al opțiunilor (pentru toți clienții);
- pe următoarele linii câte o opțiune având formatul *client*, *termen*, *sumă* unde *client* este un număr din intervalul [1, n], iar *termen* este o valoarea mai mică decât 100.

Seturile de date din fișier vor fi separate printr-o linie vidă.

Date de ieșire

Pentru fiecare set de date va trebui să scrieți la ieșirea standard câte o linie pe care se va afla profitul maxim care poate fi obținut. Datele de ieșire corespunzătoare unor seturi de date consecutive vor fi separate printr-o linie vidă.

Exemplu

C.DAT	*Ieșire*
3	26
2	
2	
4	
4	
1 2 14	
3 4 25	
2 4 12	
3 3 10	

P080204: Galerie de artă

Galeriile de artă din noua clădire a *Centrului pentru Cooperare Balcanică* au formă poligonală (nu neapărat convexă).

Când are loc o expoziție de anvergură, apare problema supravegherii tuturor tablourilor expuse. Va trebui să scrieți un program care, pentru o galerie dată, determină aria suprafeței din care pot fi observate toate punctele de pe pereții galeriei.

În figura 1 este prezentată o hartă a unei galerii, iar în figura 2 este hașurată suprafața cerută.

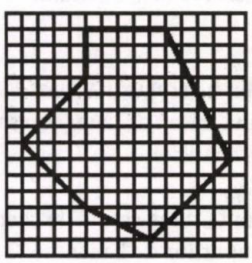


Figura 1

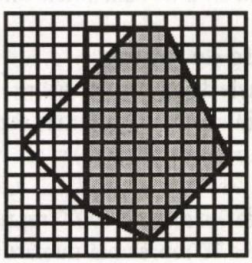


Figura 2

Date de intrare

Prima linie a fișierului de intrare **D.DAT** conține numărul T al seturilor de date. Fiecare set de date conține o linie pe care se află numărul N ($5 \le N \le 1500$) al vârfurilor poligonului și N linii care conțin câte două numere întregi care reprezintă coordonatele unuia dintre vârfuri. Coordonatele pot fi reprezentate ca numere întregi pe 16 biți și sunt separate printr-un singur spațiu.

Date de ieșire

Pentru fiecare set de date va trebui să scrieți la ieșirea standard câte o linie pe care se va afla aria suprafeței cerute (valoarea va fi scrisă folosind două zecimale exacte).

Exemplu

D.DAT	*Ieșire*
1	80.00
7	
0 0	
4 4	
4 7	
9 7	
13 -1	
8 -6	
4 -4	

P080205: Joc

Recent a avut loc un campionat internațional de volei. Suntem interesați de probabilitatea de a câștiga partida pe care o are fiecare dintre cele două echipe în condițiile în care avem la dispoziție anumite informații despre echipe. Vom prezenta regulile jocului și informațiile necesare:

- Partida se desfășoară între două echipe *A* și *B*.
- Prima echipă care câștigă *K* seturi, câștigă partida.
- Un set constă în mai multe runde; fiecare rundă este câștigată de o echipă; pentru fiecare rundă câștigată echipa primește un punct.
- Setul este câștigat de echipa care acumulează *L* puncte.
- Dacă echipa *A* se află la serviciu, atunci există o probabilitate de *Pa*% să câștige runda și o probabilitate de (100 - *Pa*)% să o piardă.
- Dacă echipa *B* se află la serviciu, atunci există o probabilitate de *Pb*% să câștige runda și o probabilitate de (100 - *Pb*)% să o piardă.
- Dacă o rundă nu este prima dintr-un set, atunci servește echipa care a câștigat runda precedentă.
- Dacă un set nu este primul dintr-o partidă, atunci la prima rundă a setului respectiv va servi echipa care nu a servit la prima rundă a setului anterior.
- Cele două echipe au șanse egale de a servi în prima rundă a primului set.

Așadar, va trebui să determinați probabilitatea, exprimată în procente, ca echipa *A* să câștige partida.

Date de intrare

Prima linie a fișierului de intrare **E.DAT** conține numărul seturilor de date. Pentru fiecare set de date fișierul de intrare va conține o linie pe care se vor afla numerele întregi *Pa*, *Pb*, *K* și *L* ($1 \le K \le 100$, $1 \le L \le 100$).

Date de ieșire

Pentru fiecare set de date va trebui să scrieți la ieșirea standard câte o linie care va conține un număr (scris cu o zecimală exactă), reprezentând șansele de câștig ale echipei A.

Exemplu

E.DAT	*Ieșire*
2	93.8
100 50 1 3	99.5
100 1 1 1	

P080206: Proces server

Una dintre paradigmele procesării paralele este paradigma producător / consumator. Ea poate fi implementată folosind un proces server și mai multe procese client. Procesele client pot reprezenta producători sau consumatori. Procesul server păstrează informații referitoare la procesele client. Fiecare proces este identificat printr-un cost care este un număr cuprins între 1 și 10000. Nu pot exista mai mult de 10000 de procese care să aibă același cost. Coada este implementată în așa fel încât să poată satisface trei tipuri de cereri:

- *a x* - se adaugă în coadă un proces cu costul *x*;
- *r* - un proces este eliminat din coadă (dacă este posibil) în concordanță cu politica de eliminare actuală;
- *p i* - stabilește politica *i* pentru server, unde *i* poate avea una dintre valorile 1 și 2. Politica implicită este 1;
- *e* - indică sfârșitul listei de cereri.

Există două politici de eliminare:

- 1 - se elimină procesul cu cel mai mic cost;
- 2 - se elimină procesul cu cel mai mare cost.

Serverul va afișa costul unui proces eliminat numai dacă numărul de ordine al procesului eliminat se află în lista de eliminare.

Va trebui să scrieți un program care simulează procesul server.

Date de intrare

Datele de intrare vor fi citite din fișierul text **F.DAT**. Fiecare set de date din acest fișier va avea următorul format:

- costul maxim al proceselor;
- lungimea listei de eliminare;
- lista de eliminare - o listă care conține numere de ordine ale proceselor care vor fi afișate; de exemplu, lista 1 4 indică faptul că vor fi scrise costurile primului și celui de al patrulea proces eliminat;
- lista cererilor care conține câte o cerere pe fiecare linie.

Fiecare set de date se încheie printr-o cerere *e*, iar seturile de date sunt separate prin linii vide.

Date de ieșire

Programul vostru va scrie la ieșirea standard costul fiecărui proces eliminat dacă numărul de ordine al cererii de eliminare se află în lista de eliminare și coada nu este vidă în momentul respectiv (dacă este vidă se va scrie valoarea -1). Fiecare cost va fi scris pe câte o linie, iar costurile corespunzătoare unor seturi de date consecutive vor fi separate printr-o linie vidă.

Exemplu

F.DAT	*Ieșire*
5	2
2	5
1 3	
a 2	
a 3	
r	

```
a 4
p 2
r
a 5
r
e
```

P080207: Rețea

Vi s-a dat sarcina să proiectați o rețea care să lege anumite puncte dintr-o zonă extinsă. Se dau punctele din zonă și o mulțime de trasee posibile pentru cablurile care pot conecta perechi de puncte. Pentru fiecare traseu posibil dintre două puncte se cunoaște lungimea cablului necesar conectării celor două puncte folosind traseul respectiv. Între două puncte pot exista mai multe trasee. Se presupune că folosind traseele date se pot interconecta (direct sau indirect) oricare două puncte.

Va trebui să proiectați rețeaua astfel încât să existe o conexiune (directă sau indirectă) între oricare două puncte ale rețelei (toate punctele sunt interconectate, dar nu neapărat prin cabluri directe) și costul total al cablurilor folosite este minim.

Date de intrare

Fișierul de intrare **G.DAT** conține mai multe seturi de date, fiecare dintre ele caracterizând o zonă. Prima linie dintr-un set de date conține numărul P al punctelor și numărul R al traseelor posibile. Următoarele R linii conțin câte trei numere; primele două identifică cele două puncte unite prin traseu, iar al treilea reprezintă lungimea traseului. Numerele sunt separate prin spații. Ultimul set de date conține doar valoarea $P = 0$.

Există cel mult 50 de puncte, iar lungimea maximă a unui traseu este 100. Numărul traseelor posibile este nelimitat. Punctele sunt identificate prin numere întregi cuprinse între 1 și P. Traseele dintre două puncte i și j pot fi date fie prin i j, fie prin j i.

Date de ieșire

Pentru fiecare set de date va trebui să scrieți la ieșirea standard câte o linie care va conține lungimea totală a cablurilor folosite pentru interconectare.

Exemplu

G.DAT	*Ieșire*
1 0	0
	17
2 3	16
1 2 37	26
2 1 17	
1 2 68	
3 7	
1 2 19	
2 3 11	
3 1 7	
1 3 5	
2 3 89	
3 1 91	
1 2 32	
5 7	
1 2 5	
2 3 7	
2 4 8	
4 5 11	
3 5 10	
1 5 6	
4 2 12	
0	

P080208: Număr șmecher

Vom numi număr șmecher un șir A care conține N numere întregi din mulțimea {0,1,2} (de exemplu, $A_0 = 1$, $A_1 = 1$, $A_2 = 0$ și $A_3 = 2$). Un număr șmecher este numit **UNU** dacă $A_0 = 1$ și $A_I = 0$ pentru $I = 1, 2, \ldots, N - 1$.

Considerăm următoarea operație dintre două numere șmechere, numită înmulțire stelată:

$$C_k = \sum_{i=0}^{k} A_i \cdot B_{k-i} + \sum_{i=k+1}^{N-1} A_i \cdot B_{N+k-i}.$$

Rezultatul operației este un șir C care nu este neapărat un număr șmecher. Această operație va fi reprezentată prin simbolul "*". Mai mult, avem la dispoziție și o operație modulo care poate fi aplicată rezultatului unei înmulțiri stelate: $(C \bmod Q)_i = C_i \bmod Q$, unde Q este un număr întreg pozitiv.

Se dă un număr șmecher A și o valoare Q. Va trebui să determinați numărul șmecher invers B. Acesta satisface relația $(A * B) \bmod Q = \mathbf{UNU}$.

Date de intrare

Prima linie a fișierului de intrare **H.DAT** conține numărul K al seturilor de date. Fișierul conține două linii pentru fiecare test. Prima dintre acestea conține numerele întregi N ($5 \le N \le 50$) și Q ($2 \le Q \le 100$), separate prin spații. Cea de-a doua linie conține N numere întregi din mulțimea {0, 1, 2} care reprezintă numărul șmecher A.

Date de ieșire

Pentru fiecare dintre cele K seturi de date se va scrie câte o linie la ieșirea standard. Dacă există un număr șmecher invers, linia va conține mesajul "`A solution can be found`", iar în caz contrar se va scrie "`No solution`".

Exemplu

H.DAT	*Ieșire*
2	`A solution can be found`
2 5	`No solution`
1 0 1 0 1	
65 8	
1 2 2 2 1 1 2 2	

AMINTIRI

În numărul trecut v-am prezentat enunțurile a șase probleme care au fost propuse spre rezolvare elevilor de clasa a IX-a care au participat la olimpiadele județene de acum opt ani. Acum ne vom aminti cum arătau în acea perioadă problemele pentru clasa a X-a. Evident, am "transformat" din nou enunțurile pentru a respecta standardele actuale.

P080209: Culori

Olimpiada Județeană de Informatică, clasa a X-a, Botoșani 1994

Se consideră o matrice de dimensiuni $m \times n$, ale cărei elemente sunt cifre cuprinse între 1 și 9, reprezentând culori. Definim ***mulțimea conexă*** a unui element ca fiind totalitatea elementelor în care se poate ajunge pornind de la elementul considerat, deplasându-ne la fiecare pas cu câte o poziție în una dintre cele patru direcții (sus, jos, stânga, dreapta), respectând condiția ca elementul în care ajungem să aibă aceeași culoare cu elementul de la care am plecat. Va trebui să determinăm, pentru fiecare culoare, numărul maxim de elemente pe care îl are o mulțime conexă ale cărei elemente au culoarea respectivă.

Date de intrare

Prima linie a fișierului de intrare **CULORI.IN** va conține dimensiunile m și n ale matricei, separate prin câte un spațiu. Fiecare dintre următoarele m linii va conține câte n cifre, separate prin spații; cifrele de pe aceste linii reprezintă elementele matricei.

Date de ieșire

Fișierul de ieșire **CULORI.OUT** va conține nouă linii. Cea de-a i-a linie a acestui fișier va conține cardinalul maxim al unei mulțimi conexe ale cărei elemente au culoarea i.

Restricție

- $1 \le m, n \le 100$.

Exemple

CULORI.IN

```
6 5
1 1 1 1 2
3 1 1 2 2
3 3 1 1 2
4 5 6 7 7
4 5 6 7 1
5 8 8 8 1
```

CULORI.OUT

```
6
4
3
2
2
2
3
3
0
```

Timp de execuție: 1 secundă/test

P080210: Regiune

Olimpiada Județeană de Informatică, clasa a X-a, Cluj 1994

Se consideră n puncte distincte în plan, având coordonate reale. Fiecare punct se unește prin segmente cu toate punctele aflate la distanță minimă față de el. Se numește ***regiune*** o mulțime maximală de puncte cu proprietatea că oricare două dintre ele sunt unite printr-un drum (succesiune de segmente adiacente). Să se scrie un program care determină toate regiunile și punctele din care sunt formate acestea.

Date de intrare

Fișierul de intrare **REGIUNE.IN** conține pe prima linie numărul n al punctelor din plan. Fiecare dintre următoarele n linii conține câte două numere reale, separate prin spații, reprezentând coordonatele unuia dintre puncte.

Date de ieșire

Prima linie a fișierului de ieșire **REGIUNE.OUT** va conține numărul k al regiunilor determinate. Pe fiecare dintre următoarele k linii este descrisă câte o regiune. Primul număr de pe una dintre aceste linii reprezintă numărul n_i al punctelor din care este formată o regiune. Următoarele n_i numere de pe linie reprezintă numerele de ordine ale punctelor care fac parte din regiunea respectivă. Ordinea este cea în care apar punctele în fișierul de intrare, iar numerotarea

începe cu 1. Numerele de pe o linie sunt separate prin spații.

Resztricție
- $2 \le n \le 30$.

Exemplu

REGIUNE.IN
```
8
1 1
3 1
1 3
3 3
4 4
5 3
7 4
7 5
```

REGIUNE.OUT
```
2
6 1 2 3 4 5 6
2 7 8
```

Timp de execuție: 1 secundă/test

P070211: Piramida

Olimpiada Județeană de Informatică, clasa a X-a
Dolj 1994

Se consideră n cuvinte ordonate lexicografic pe baza unui alfabet neobișnuit A. Cuvintele sunt formate din litere ale alfabetului englezesc, dar ordinea lexicografică pentru alfabetul A nu este neapărat cea din alfabetul englezesc. Să se determine, dacă este posibil, o ordine în care ar putea apărea literele în alfabetul A.

Date de intrare

Fișierul de intrare **ALFABET.IN** va conține pe prima linie numărul n al cuvintelor. Fiecare dintre următoarele n linii conține câte un cuvânt. Cele n cuvinte sunt ordonate lexicografic în funcție de alfabetul A.

Date de ieșire

Fișierul de ieșire **ALFABET.OUT** va conține pe prima linie numărul k al literelor alfabetului A (acesta este numărul literelor distincte care apar în fișierul de intrare). Pe a doua linie se vor afla cele k litere ale alfabetului A, separate prin câte un spațiu. Aceste litere trebuie să fie ordonate lexicografic în funcție de alfabetul A. În cazul în care nu se poate determina o ordine pentru literele alfabetului A, astfel încât cuvintele din fișierul de intrare să fie ordonate lexicografic, fișierul de ieșire va conține o singură linie pe care se va afla textul IMPOSIBIL.

Restricții și precizări
- $2 \le n \le 100$;
- un cuvânt conține cel mult 100 de litere.

Exemplu

ALFABET.IN
```
10
ginfo
agora
aranjare
metode
media
maestru
margine
internet
rezolvare
rest
```

ALFABET.OUT
```
17
g e a m i r t d z s n f o j u l v
```

Timp de execuție: 1 secundă/test

P070212: Matrice

Olimpiada Județeană de Informatică, clasa a X-a,
Mehedinți 1994

Se consideră o matrice pătratică de dimensiune n ale cărei elemente sunt numere întregi.

Matricea A este ***simetrică față de diagonala principală*** dacă pentru toate perechile (i, j) pentru care $1 \le i, j \le n$ avem $a_{ij} = a_{ji}$.

Matricea A este ***superior triunghiulară*** dacă toate elementele de sub diagonala principală sunt nule, și ***inferior triunghiulară*** dacă toate elementele de deasupra diagonalei principale sunt nule.

Va trebui să scrieți un program care verifică dacă o matrice pătratică dată este simetrică față de diagonala principală, este superior triunghiulară, inferior triunghiulară sau are toate elementele nule.

Date de intrare

Fișierul de intrare **MATRICE.IN** conține pe prima linie dimensiunea n a matricei. Fiecare dintre următoarele n linii va conține câte n numere, separate prin spații; numerele de pe aceste linii reprezintă elementele matricei.

Date de ieșire

Fișierul de ieșire **MATRICE.OUT** va conține o singură linie pe care se va afla un singur număr x. În cazul în care matricea are toate elementele nule, valoarea numărului x va fi 1. Dacă există elemente nenule în matrice și matricea este simetrică față de diagonala principală, atunci valoarea numărului x va fi 2. Dacă există elemente nenule în matrice și matricea nu este simetrică față de diagonala principală, dar este superior triunghiulară, atunci valoarea numărului x va fi 3. Dacă există elemente nenule în matrice, matricea nu este simetrică față de diagonala principală, nu este superior triunghiulară, dar este inferior triunghiulară, atunci valoarea numărului x va fi 4. În toate celelalte cazuri, valoarea numărului x va fi 5.

Restricții și precizări

- $2 \le n \le 100$;
- elementele matricei au valori cuprinse între 0 și 1000.

Exemple

MATRICE.IN
```
2
0 0
0 0
```
MATRICE.OUT
```
1
```

MATRICE.IN
```
3
1 0 0
0 1 0
0 0 1
```
MATRICE.OUT
```
2
```

MATRICE.IN
```
3
1 2 3
2 1 2
3 2 1
```
MATRICE.OUT
```
2
```

MATRICE.IN
```
3
1 2 3
0 1 4
0 0 1
```
MATRICE.OUT
```
3
```

MATRICE.IN
```
2
1 0
2 1
```
MATRICE.OUT
```
4
```

MATRICE.IN
```
3
1 0 3
0 1 0
2 0 1
```
MATRICE.OUT
```
5
```

Timp de execuție: 1 secundă/test

P070213: Vector

Olimpiada Județeană de Informatică, clasa a X-a, Mehedinți 1994

Fie V un vector care conține n elemente întregi. Să se determine o secvență de valori situate pe poziții consecutive al căror produs este maxim.

Date de intrare

Prima linie a fișierului de intrare **VECTOR.IN** conține numărul n al elementelor din vector. Următoarea linie va conține cele n elemente ale vectorului, separate prin spații.

Date de ieșire

Fișierul de ieșire **VECTOR.OUT** va conține o singură linie pe care se vor afla două numere întregi, separate prin spații. Primul dintre aceste două numere reprezintă poziția la care începe secvența, iar cel de-al doilea număr reprezintă poziția la care se termină secvența.

Restricții și precizări

- $7 \le n \le 1000000$;
- elementele vectorului au valori cuprinse între -1000000 și 1000000.

Exemple

VECTOR.IN
```
15
5 0 -2 5 -3 7 -1 1 0 8 7 -1 0 2 3
```
VECTOR.OUT
```
3 6
```

VECTOR.IN
```
5
5 -3 0 8 -1
```
VECTOR.OUT
```
4 4
```

Timp de execuție: 10 secunde/test

P070214: Sumă

Olimpiada Județeană de Informatică, clasa a X-a, Timiș 1994

Se consideră un număr întreg strict pozitiv n. Să se determine toate posibilitățile de a scrie valoarea n ca sumă de numere întregi consecutive.

Date de intrare

Fișierul de intrare **SUMA.IN** va conține un singur număr care reprezintă valoarea n.

Date de ieșire

Prima linie a fișierului de ieșire **SUMA.OUT** va conține numărul k al posibilităților de a scrie valoarea n ca sumă de numere întregi consecutive. Fiecare dintre următoarele k linii va descrie o secvență. O secvență va fi descrisă prin numărul de elemente din componența sa și prin primul element al secvenței.

Restricție

- $1 \le n \le 2000000000$.

Exemplu

SUMA.IN
```
6
```
SUMA.OUT
```
3
3 1
4 0
12 -5
```

Timp de execuție: 10 secunde/test

Yong-In, Coreea de Sud

IOI 2002

În continuare vă vom prezenta soluțiile oficiale ale celor șase probleme propuse spre rezolvare la ediția din acest an a Olimpiadei Internaționale de Informatică, desfășurată în perioada 18-25 august 2002.

P060213: Broscuțele buclucașe

Un algoritm naiv de rezolvare a acestei probleme are ordinul de complexitate $O(N^3)$ și implică parcurgerea tuturor celor $O(N^2)$ segmente care au ca extremități puncte din mulțimea S. Un segment de lungime L poate fi extins în oricare din cele două direcții ale sale cu un alt segment de lungime L, dacă extremitățile noului segment fac parte din mulțimea S. Se obține astfel un segment cu lungimea $2 \cdot L$ care poate fi extins cu un alt segment de lungime L, dacă se respectă aceeași condiție. Procedeul continuă până în momentul în care segmentul nu mai poate fi extins.

Numărul plantelor culcate la pământ de o broscuță care parcurge un astfel de segment extins va fi dat de numărul extinderilor la care se adaugă valoarea 2 (cele două plante corespunzătoare segmentului inițial). Dacă un segment nu poate fi extins până în afara lanului, atunci el nu este luat în considerare. Operația de extindere are ordinul de complexitate $O(N)$. Soluția va fi dată de numărul maxim de plante culcate la pământ corespunzătoare unui anumit segment.

Un algoritm eficient, cu ordinul de complexitate $O(N^2)$ se bazează pe algoritmul de determinare a unei ***submulțimi coliniare egal spațiate***. Algoritmul funcționează prin suprapunerea tuturor tripletelor egal spațiate cu scopul de a determina toate mulțimile coliniare egal spațiate maximale. Suprapunerea este realizată prin construirea unui graf neorientat unde, pentru fiecare triplet egal spațiat (p_A, p_B, p_C) sunt introduse nodurile $<A, B>$ și $<B, C>$ și muchia $(<A, B>, <B, C>)$; componentele conexe ale acestui graf corespund submulțimilor coliniare egal spațiate maximale din mulțimea S. Se observă că drumul unei broscuțe este un lanț liniar format din noduri conectate (cu cel puțin o muchie și două noduri, adică cel puțin trei plante culcate la pământ) ale acestui graf. Fiecare nod al grafului are gradul cel mult 2, deci mulțimea muchiilor și cea a vârfurilor grafului au ambele dimensiunea $O(N^2)$, așadar putem determina toate submulțimile coliniare egal spațiate maximale într-un timp de ordinul $O(N^2)$.

Singurul detaliu rămas în discuție este modul în care pot fi determinate eficient tripletele egal spațiate folosite în construirea grafului. O metodă evidentă este parcurgerea tuturor tripletelor, dar aceasta ar face ca ordinul de complexitate să devină $O(N^3)$. Dacă păstrăm în memorie lanul de orez sub forma unei matrice (fiecare plantă este un element al acesteia) care conține numărul de ordine al aterizării pe acea plantă (de exemplu, dacă cea de-a 100-a plantă culcată la pământ are coordonatele (10, 12), atunci valoarea elementului corespunzător poziției (10, 12) va fi 100), putem lua în considerare toate perechile de plante (p_A, p_B) și apoi determinăm în timp constant poziția plantei p_C, deoarece locația acesteia este determinată în mod unic de p_A și p_B. Ordinul de complexitate al acestei operații este $O(N^2)$, dar avem nevoie de un spațiu de memorie de ordinul $O(dimensiunea\ lanului)$. În cazul cel mai defavorabil, matricea va avea $5000 \cdot 5000$ elemente, fiecare fiind un întreg reprezentat pe 2 bytes. Așadar, această matrice va ocupa aproximativ 50 MB memorie. Datorită faptului că stocarea grafului folosit pentru rezolvare necesită, de asemenea, un spațiu de memorie de ordinul $O(N^2)$, limita de 64 MB, impusă în enunțul problemei, este depășită. Totuși, această matrice este foarte rarefiată (majoritatea elementelor au valoarea 0), deci poate fi stocată sub forma unei tabele de dispersie. Din nefericire, folosirea unei tabele de dispersie afectează ordinul de complexitate al algoritmului. Cea de-a treia și cea mai bună opțiune este construirea grafului în timp liniar și folosirea unei memorii cu dimensiune constantă prin sortarea locațiilor (în funcție de linie și în cadrul aceleiași linii în funcție de coloană) și păstrarea a trei referințe în lista obținută (referințele A, B, C vor indica punctele p_A, p_B, p_C - $A < B < C$). Construirea grafului se realizează astfel: se iau în considerare toate valorile posibile pentru A și apoi, la fiecare pas se incrementează fie valoarea B, fie valoarea C, astfel încât să fie menținută o spațiere cât mai apropiată între p_A, p_B și p_C; când spațierea este egală, nodurile și muchia corespunzătoare sunt introduse în graf.

Există de asemenea un algoritm cu ordinul de complexitate $O(N^2)$ care folosește *metoda programării dinamice*. Acest algoritm se confruntă cu aceleași probleme de memorie ca și cele descrise anterior. Pe lângă matricea folosită

în algoritmul anterior se stochează o altă matrice de dimensiune $O(N^2)$ ale cărei linii sunt indexate de p_A; fiecare linie conține N elemente, unul pentru fiecare plantă p_B, ale căror valori reprezintă numărul de aterizări dintr-un traseu al unei broscuțe care trece prin p_A și p_B, iar celelalte puncte se află înaintea lui p_A în lista ordonată. Cu alte cuvinte, presupunem că lanul se termină în poziția p_A și căutăm trasee ale broscuțelor (care pot avea orice lungime) în acest lan de dimensiune mai mică. Ideea este de a determina trasee parțiale ale broscuțelor care nu încalcă nici una dintre condițiile referitoare la trasee în regiunea deja examinată a lanului. Presupunând că primele A linii ale matricei sunt completate, a $(A + 1)$-a linie este completată luând în considerare toate cele $O(N)$ plante culcate la pământ B, care se află în listă înainte de p_A și, dacă există o plantă culcată la pământ C, astfel încât A, B și C sunt egal spațiate, se caută în șir numărul aterizărilor din traseul de la C la B, se incrementează cu 1 și se stochează ca al B-lea element de pe linia A. Dacă C este în afara lanului, atunci valoarea corespunzătoare va fi 2. În același timp, se verifică dacă următoarea plantă culcată la pământ (B) se află în afara grafului; în acest caz a fost determinat un traseu complet. Pentru determinarea eficientă a valorii C avem nevoie de aceeași matrice de 50 MB, descrisă pentru algoritmul anterior. Din nou, poate fi folosită o tabelă de dispersie, dar în cazul cel mai defavorabil, ordinul de complexitate al algoritmului crește.

P060214: Utopia divizată

Aceasta este o problemă în spațiul bidimensional (plan), dar este mai ușor de rezolvat dacă o considerăm ca două probleme într-un spațiu unidimensional. Pentru început vom arăta cum poate fi rezolvată problema mai simplă, într-un spațiu unidimensional. Enunțul acestei probleme ar fi:

Dându-se N numere de cod naturale și o secvență de N semne corespunzătoare celor două regiuni ('+' sau '-'), determinați o secvență de valori de cod cu semn x, astfel încât $\sum_{i=1}^{k} x_i$ *are semnul identic cu cel corespunzător celei de a k-a regiuni.*

Soluția este intuitivă, dar demonstrarea corectitudinii acesteia este puțin mai dificilă. La început, vom sorta în ordine crescătoare cele N numere de cod și apoi le vom atribui semne alternante. Cu alte cuvinte, pentru $|x_i| > |x_i - 1|$ avem $x_i > 0$ dacă și numai dacă $x_{i-1} < 0$. Semnul atribuit celui mai mic cod determină semnele atribuite tuturor celorlalte coduri. Vom începe dintr-o poziție aflată în "mijlocul" secvenței ordonate și ne vom deplasa înspre exterior. Vom folosi numere mai mici pentru a nu modifica semnul sumei și numere mai mari pentru a-l modifica. Pentru a clarifica faptul că acest algoritm este corect, vom prezenta câteva rezultate matematice.

Definiție

Vom numi ***secvență alternantă*** o secvență $X = (x_a, x_{a+1}, ..., x_b)$, $a \le b$ pentru care:

- $|x_a| < |x_{a+1}| < |x_{a+2}| < ... < |x_b|$ și
- pentru fiecare valoare i $(a < i \le b)$, semnele valorilor x_i și x_{i-1} sunt diferite.

Lemă

Fie $X = (x_a, x_{a+1}, ..., x_b)$, $a \le b$ o secvență alternantă. Semnul valorii x_b este același cu semnul sumei tuturor elementelor din X.

Demonstrație

Vom presupune fără a reduce generalitatea că x_b este un număr întreg pozitiv. Dacă numărul elementelor din X $(b - a + 1)$ este par, atunci sumele parțiale $x_a + x_{a+1}$, $x_{a+2} + x_{a+3}$, ..., $x_{b-1} + x_b$ sunt pozitive, deci valoarea sumei tuturor elementelor din X este pozitivă. Dacă numărul elementelor din X este impar, atunci sumele parțiale x_a, $x_{a+1} + x_{a+2}$, ..., $x_{b-1} + x_b$ sunt pozitive, deci valoarea sumei tuturor elementelor din X este din nou pozitivă.

Cazul în care valoarea x_b este negativă se tratează într-o manieră asemănătoare (toate sumele parțiale vor fi negative, deci suma tuturor elementelor va fi negativă).

Exemple

Suma elementelor din secvența alternantă X = (-1, +2, -5, +6) este egală cu (-1 + 2) + (-5 + 6) = +2 și este pozitivă la fel ca ultimul element al secvenței.

Suma elementelor din secvența alternantă X = (+3, -4, +5, -6, +7) este egală cu (+3) + (-4 + 5) + (-6 + 7) = +5 și este pozitivă la fel ca ultimul element al secvenței.

Fie $S = (s_a, s_{a+1}, ..., s_b)$, $a \le b$ o secvență de semne. O secvență $X'=(x'_a, x'_{a+1}, ..., x'_b)$, $a \le b$ este validă pentru S, dacă semnul sumei $\sum_{i=a}^{k} x'_i$ este s_k pentru fiecare k cuprins între a și b.

Teoremă

Fie $(x_a, x_{a+1}, ..., x_b)$, $a \le b$ o secvență alternantă și $S = (s_a, s_{a+1}, ..., s_b)$, $a \le b$ o secvență de semne. Dacă semnul valorii x_b este s_b, atunci există o secvență $X'=(x'_a, x'_{a+1}, ..., x'_b)$, $a \le b$ astfel încât:

- $\{x'_a, x'_{a+1}, ..., x'_b\}=\{x_a, x_{a+1}, ..., x_b\}$ și
- X' este validă pentru S.

Demonstrație

Demonstrația se realizează pe baza unei inducții matematice după numărul k al elementelor din X. Pentru $k = 1$ se observă foarte ușor că $X' = X$ este o secvență validă pentru S. Presupunem că $k \ge 2$ și notăm $S_1 = S - s_b$. Cu alte cuvinte, avem $S_1=(s_a, s_{a+1}, ..., s_{b-1})$.

Distingem două cazuri. Dacă semnul valorii x_b este s_b, atunci notăm $X_1 = X - x_a$ și $t = x_a$. Dacă semnele sunt diferite atunci avem $X_1 = X - x_b$ și $t = x_b$.

Se observă că X_1 este o secvență alternantă și S_1 este o secvență de semne, astfel încât semnul ultimului element din X_1 este s_{b-1} (ultimul element din S_1). Așadar, folosind ipoteza de inducție, deducem că există o secvență X'_1

validă pentru S_1. Prin urmare, $X' = (X'_1, t)$ este o secvență validă pentru S.

Exemple

Pentru secvența alternantă $X = (-4, +5, -7, +8)$ și secvența de semne $S = (+, -, -, +)$ avem:

$S_1 = (+, -, -)$, $X_1 = (-4, +5, -7)$

$S_2 = (+, -)$, $X_2 = (+5, -7)$

$S_3 = (+)$, $X_3 = (+5)$

Ca urmare, obținem:

$X'_3 = (+5)$

$X'_2 = (+5, -7)$

$X'_1 = (+5, -7, -4)$

$X' = (+5, -7, -4, +8)$

Pentru secvența alternantă $X = (-1, +2, -3)$ și secvența de semne $S = (-, -, -)$ avem:

$S_1 = (-, -)$, $X_1 = (+2, -3)$

$S_2 = (-)$, $X_2 = (-3)$

Ca urmare, obținem:

$X'_2 = (-3)$

$X'_1 = (-3, +2)$

$X' = (-3, +2, -1)$

Acum putem prezenta pașii algoritmului de rezolvare a acestei probleme:

Pasul 1: *//citirea datelor de intrare*

1.1. citirea valorii N.

1.2. citirea celor $2 \cdot N$ numere de cod și partiționarea lor în două mulțimi A și B cu același cardinal.

1.3. citirea secvenței de regiuni $R = (r_1, r_2, ..., r_N)$.

Pasul 2: *//determinarea coordonatelor orizontale ale perechilor de coduri*

2.1. determinarea unei secvențe de semne $S = (s_1, s_2, ..., s_N)$ astfel încât pentru toate valorile j cuprinse între 1 și N avem s_j = '+' dacă $r_j = 1$ sau $r_j = 4$ și s_j = '-' dacă $r_j = 2$ sau $r_j = 3$.

2.2. determinarea unei secvențe alternante $X = (x_1, x_2, ..., x_N)$ construită pe baza mulțimii A, astfel încât semnul valorii x_N este s_N.

2.3. determinarea, pe baza teoremei anterioare, a unei secvențe $X' = (x'_1, x'_2, ..., x'_N)$ care este validă pentru S.

Pasul 3: *//determinarea coordonatelor verticale ale perechilor de coduri*

3.1. determinarea unei secvențe de semne $S = (s_1, s_2, ..., s_N)$ astfel încât pentru toate valorile j cuprinse între 1 și N avem s_j = '+' dacă $r_j = 1$ sau $r_j = 2$ și s_j = '-' dacă $r_j = 3$ sau $r_j = 4$

3.2. determinarea unei secvențe alternante $Y = (y_1, y_2, ..., y_N)$ construită pe baza mulțimii B, astfel încât semnul valorii y_N este s_N.

3.3. determinarea unei secvențe $Y' = (y'_1, y'_2, ..., y'_N)$ care este validă pentru S.

Pasul 4: *//scrierea datelor de ieșire*

Afișarea perechilor $(x'_1, y'_1), (x'_2, y'_2), ..., (x'_N, y'_N)$.

Teoremă

Algoritmul prezentat anterior este corect și are ordinul de complexitate $O(N \cdot \log N)$.

Demonstrație

Corectitudinea algoritmului rezultă direct din teorema anterioară. Ordinul de complexitate al tuturor pașilor (excepție 2.2 și 3.2) este $O(N)$. Pașii 2.2 și 3.2 au ordinul de complexitate $O(N \cdot \log N)$ datorită faptului că necesită efectuarea unei sortări.

Observație

Nu se cunoaște o soluție cu un ordin de complexitate mai mic, dar nici nu s-a reușit demonstrarea faptului că sortarea este neapărat necesară pentru rezolvarea acestei probleme.

P060215: Compresie XOR

Vom începe prezentarea soluției acestei probleme definind anumiți termeni pe care îi vom folosi ulterior.

Definiții

Grila corespunzătoare ecranului este grila obținută prin trasarea unor linii imaginare care separă pixelii.

Un ***punct al grilei*** este punctul de intersecție a două linii ale grilei corespunzătoare ecranului (o linie verticală și una orizontală).

Un ***colț*** este un punct al grilei care este adiacent cu un număr impar de pixeli negri. Dacă un punct p este colț, spunem că ***ic-valoarea*** sa este 1 și notăm acest fapt prin $ic(p) = 1$, iar dacă p nu este colț, spunem că *ic-valoarea* sa este 0 și $ic(p) = 0$.

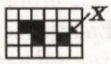


Figura 1

De exemplu, în figura 1, punctul x este adiacent cu patru pixeli dintre care 1 este negru și 3 sunt albi. Așadar, punctul x este colț. În figura 1 avem în total 10 colțuri.

Este evident că următoarea afirmație este adevărată: ***O imagine este complet albă dacă și numai dacă ea nu conține nici un colț.***

Putem acum să încercăm să găsim o soluție folosind imaginea furnizată ca intrare și încercând să obținem imaginea complet albă cu ajutorul operațiilor XOR. Această soluție se bazează pe următoarea lemă.

Lemă

Presupunem că efectuăm operația $XOR(L, R, T, B)$; fie Q dreptunghiul afectat de această operație. Vom nota prin a punctul din colțul stânga-sus al dreptunghiului Q, prin b punctul din colțul stânga-jos, prin c punctul din colțul dreapta-jos și prin d punctul din colțul dreapta-sus. Prin efectuarea operației $XOR(L, R, T, B)$ *ic-valorile* punctelor a, b, c și d se modifică (dacă au fost 0 devin 1 și invers), iar

ic-valorile tuturor celorlalte puncte ale grilei rămân neschimbate.

Demonstrație
În urma operației, exact un pixel adiacent fiecăruia dintre punctele *a*, *b*, *c* și *d* își schimbă culoarea. Ca urmare, *ic-valorile* acestor puncte se modifică.

Luăm acum în considerare punctele de pe laturile dreptunghiului *Q*. Pentru fiecare dintre acestea, exact doi pixeli adiacenți își schimbă culoarea (cei din interiorul dreptunghiului), deci paritatea numărului de pixeli negri din jurul acestor puncte nu se modifică.

În final considerăm punctele din interiorul dreptunghiului *Q*. Toți cei patru pixeli adiacenți cu aceste puncte își schimbă culoarea, deci paritatea numărului de pixeli negri adiacenți rămâne aceeași.

Vom încerca să reducem numărul de colțuri la fiecare pas; algoritmul se va opri când nu va mai exista nici un colț. Evident, vom încerca să reducem cât mai mult acest număr, la fiecare pas. Totuși, nu vom putea întotdeauna să efectuăm o operație *XOR* în urma căreia să se modifice *ic-valorile* tuturor colțurilor din regiunea afectată de această operație.

Lemă
Dacă imaginea nu este complet albă, este întotdeauna posibil să efectuăm o operație *XOR*, astfel încât numărul colțurilor să se reducă fie cu 2, fie cu 4.

Demonstrație
Pentru început, trebuie să observăm că există o linie pe care se află cel puțin două colțuri. Este evident că linia cea mai de sus care conține pixeli negri conține un colț aflat în partea din dreapta-sus a celui mai din dreapta pixel negru și un colț aflat în partea din stânga-sus a celui mai din stânga pixel negru. Aceste colțuri sunt distincte și se află pe aceeași linie.

În continuare, vom observa că nu este posibil ca toate colțurile să se afle pe aceeași linie. Este evident că cel mai de sus pixel negru și cel mai de jos pixel negru au în jurul lor (deasupra, respectiv dedesubt) cel puțin câte un colț care nu se află pe aceeași linie orizontală. De asemenea, cel mai din dreapta pixel negru și cel mai din stânga pixel negru au în jurul lor (la dreapta, respectiv la stânga) cel puțin câte un colț care nu se află pe aceeași linie verticală.

Vom putea alege acum un dreptunghi determinat de trei colțuri. Mai întâi alegem cel mai din dreapta colț de pe cea mai de sus linie și apoi un alt colț care se află pe aceeași linie verticală. Cel de-al treilea colț poate fi determinat astfel: începem de la primul colț ales; acesta are un singur pixel adiacent negru (cel din stânga-jos), ceilalți fiind albi; parcurgem punctele aflate la stânga pe aceeași linie până în momentul când pixelul aflat la stânga și sub punctul curent devine alb. Aceste trei colțuri determină dreptunghiul asupra căruia vom efectua operația *XOR*.

Așadar, cel puțin trei dintre punctele care delimitează dreptunghiul sunt colțuri. Dacă toate cele patru puncte sunt colțuri, atunci în urma operației numărul total al colțurilor se reduce cu 4. Dacă cel de-al patrulea punct nu este colț, numărul total al colțurilor se reduce cu 2 (se elimină trei colțuri și se creează unu).

Așadar, un algoritm de rezolvare al acestei probleme constă în alegerea succesivă a trei colțuri care determină un dreptunghi și efectuarea de operații *XOR* asupra acestor dreptunghiuri până în momentul în care nu mai rămâne nici un pixel negru în imagine.

Teoremă
Algoritmul prezentat anterior este corect și numărul operațiilor *XOR* care trebuie efectuate este de cel mult două ori mai mare decât numărul minim de operații *XOR* care trebuie efectuate pentru eliminarea tuturor pixelilor negri.

Demonstrație
Lema anterioară ne asigură că la fiecare pas vom găsi 3 colțuri care determină un dreptunghi. Datorită faptului că numărul inițial al colțurilor este finit și la fiecare pas acesta se reduce, după un număr finit de pași imaginea nu va mai conține nici un colț, deci va fi complet albă.

În cazul ideal, pentru fiecare operație *XOR*, numărul colțurilor se reduce cu 4. În cazul algoritmului nostru, numărul colțurilor se reduce cu cel puțin 2. Așadar, în cel mai defavorabil caz, vom elimina câte două colțuri la fiecare pas. Ca urmare, numărul pașilor va fi de cel mult două ori mai mare față de cazul ideal.

Vom prezenta în continuare o versiune a acestui algoritm:

```
pentru fiecare linie i execută
  pentru fiecare coloană j execută
    dacă colț(i, j) atunci
      fie k cel mai apropiat colț aflat pe linia i la dreapta față de punctul (i, j)
      fie l cel mai apropiat colț aflat pe coloana j sub punctul (i, j)
      xor(j, k-1, i, l+1)
    sfârșit dacă
  sfârșit pentru
sfârșit pentru
```

Folosind acest algoritm se puteau obține aproximativ 85 de puncte la această problemă.

Problema poate fi abordată și folosind tehnici euristice. Una dintre acestea constă în determinarea, la fiecare pas, a unui dreptunghi care conține doar pixeli negri și efectuarea operației *XOR* asupra dreptunghiului respectiv. Punctajul obținut în acest caz este de aproximativ 35 de puncte.

O a treia abordare folosește un algoritm de determinare a unui cuplaj maxim într-un graf. Se iau în considerare

toate colțurile de pe o linie, fiecare devenind un nod al grafului. Costul unei muchii dintre două vârfuri este 1 dacă și numai dacă există un dreptunghi determinat de cele două colțuri corespunzătoare celor două vârfuri și alte două colțuri din imagine. În caz contrar, costul muchiei este 0. Se determină un cuplaj maxim în graful construit și se aplică operații *XOR* asupra tuturor dreptunghiurilor determinate de muchiile care apar în acest cuplaj. Operația continuă până în momentul în care au fost eliminați toți pixelii. Această abordare ar fi dus la obținerea tuturor celor 100 de puncte (mai exact 99.7, care prin rotunjire devine 100).

Pentru unul singur (ultimul) dintre cele zece teste prima metodă duce la obținerea unui rezultat mai bun decât a treia.

P060216: Programarea pachetelor

O posibilitate de a aborda această problemă este folosirea *programării dinamice*. Vom nota prin C_i costul minim de partiționare a operațiilor $J_i, J_{i+1}, ..., J_n$ în pachete și prin $C_i(k)$ costul minim de partiționare ale acestor operații, dacă primul pachet este format din operațiile $J_i, J_{i+1}, ..., J_k$. Vom obține următoarea relație:

$C_i(k) = C_k + (S + T_i + T_{i+1} + ... + T_{k-1}) \cdot (F_i + F_{i+1} + ... + F_n)$.

Pe baza acestei relații deducem următoarea formulă de calcul pentru determinarea valorilor C_i:

$C_i = \min\{C_i(k) \mid k = i + 1, ..., n + 1\}$ pentru $1 \le i \le n$ și $C_{n+1} = 0$.

Este evident că ordinul de complexitate al acestui algoritm este $O(n^2)$, deoarece trebuie calculate valorile $C_i(k)$ pentru toate valorile i și k astfel încât $1 \le i \le k \le n$).

Studiind valorile $C_i(k)$ se poate deduce că această problemă se poate rezolva în timp liniar după cum se va arăta în continuare.

Din relația $C_i(k) = C_k + (S + T_i + T_{i+1} + ... + T_{k-1}) \cdot (F_i + F_{i+1} + ... + F_n)$ deducem că, pentru $i < k < l$ avem:

$C_i(k) < C_i(l) \Leftrightarrow$

$\Leftrightarrow C_l - C_k + (T_i + T_{i+1} + ... + T_{k-1}) \cdot (F_i + F_{i+1} + ... + F_n) \ge 0$

$\Leftrightarrow (C_k - C_l) / (T_i + T_{i+1} + ... + T_{k-1}) \le (F_i + F_{i+1} + ... + F_n)$.

Vom nota prin $g(k, l)$ valoarea $(C_k - C_l) / (T_i + T_{i+1} + ... + T_{k-1})$ și prin $f(i)$ valoarea $(F_i + F_{i+1} + ... + F_n)$.

Proprietăți

- Dacă $g(k, l) \le f(i)$ pentru $1 \le i < k < l$, atunci $C_i(k) \le C_i(l)$.
- Dacă $g(j, k) \le g(k, l)$ pentru $1 \le i < k < l$, atunci pentru toate valorile i cuprinse între 1 și $j - 1$ avem $C_i(j) \le C_i(k)$ sau $C_i(l) \le C_i(k)$.

Din cea de-a doua proprietate rezultă că dacă avem $g(j, k) \le g(k, l)$ pentru $1 \le i < k < l \le n$, atunci valoarea $C_i(k)$ nu este necesară pentru a calcula valoarea C_i.

Vom calcula valorile C_i în ordinea $C_n, C_{n-1}, ..., C_1$. Vom folosi o structură $Q = (i_r, i_{r-1}, ..., i_2, i_1)$ asemănătoare unei cozi, ultimul element introdus (coada) fiind i_r, iar primul (capul) fiind i_1. Singura diferență față de o coadă "clasică" este faptul că pot fi eliminate elemente din ambele capete ale cozii. Această structură respectă următoarele proprietăți:

- $i_r < i_{r-1} < ... < i_2 < i_1$ și
- $g(i_r, i_{r-1}) > g(i_{r-1}, i_{r-2}) > ... > g(i_2, i_1)$.

În momentul calculării unei valori C_i va trebui, mai întâi, să eliminăm elementele care nu sunt necesare din capul cozii Q. Pentru aceasta vom folosi valoarea $f(i)$ și prima proprietate enunțată anterior. Dacă $f(i) \ge g(i_2, i_1)$ vom elimina din coadă elementul i_1, deoarece din proprietatea amintită rezultă că pentru toate valorile $h \le i$, avem $f(h) \ge f(i) \ge g(i_2, i_1)$, deci $C_h(i_2) \le C_h(i_1)$. Vom continua să eliminăm elemente până ajungem în situația în care, pentru o anumită valoare $t \ge 1$, avem $g(i_r, i_{r-1}) > g(i_{r-1}, i_{r-2}) > ... > g(i_{t+1}, i_t) > f(i)$. Acum, pe baza aceleiași proprietăți, deducem că $C_i(i_{v+1}) > C_i(i_v)$ pentru toate valorile v cuprinse între t și $r - 1$, sau $r = t$ (caz în care în coadă a rămas un singur element: i_r). Rezultă imediat că:

$C_i(i_t) = \min\{C_i(k) \mid k = i + 1, ..., n + 1\}$.

În momentul inserării elementului i în coadă, trebuie efectuate operațiile necesare pentru ca cele două proprietăți ale acesteia să fie respectate. Dacă avem $g(i, i_r) \le g(i_r, i_{r-1})$, atunci din cea de-a doua proprietate enunțată anterior, rezultă că elementul i_r trebuie eliminat. Vom continua să eliminăm elemente până ajungem în situația în care $g(i, i_r) > g(i_r, i_{r-1})$. În acest moment i este adăugat în coadă (devine ultimul element al acesteia).

Fiecare valoare i este inserată și eliminată din coadă cel mult o dată. Pentru fiecare inserare și pentru fiecare eliminare operațiile necesare sunt realizate în timp constant. Așadar, ordinul de complexitate al algoritmului prezentat este $O(n)$.

P060217: Stații de autobuz

Soluția acestei probleme se bazează pe un algoritm al cărui ordine de complexitate este $O(n^3)$. *Diametrul* unei rețele de autobuze va fi dat de cel mai lung traseu dintre două stații ale rețelei. Va trebui să găsim cel mai mic diametru posibil, considerarând toate posibilitățile de a alege stațiile centrale. Vom lua în considerare două cazuri și, pentru început, vom introduce câteva notații:

- Fie D_1 valoarea minimă a celui mai lung traseu dintre două stații care sunt legate prin intermediul unei singure stații centrale, ținând cont de toate posibilitățile de a alege stația centrală respectivă.
- Fie D_2 valoarea minimă a celui mai lung traseu dintre două stații care sunt legate prin intermediul ambelor stații centrale, luând în considerare toate posibilitățile de a alege cele două stații centrale.

Vom determina valorile D_1 și D_2 și vom furniza la ieșire minimul dintre aceste două valori.

Vom descrie acum modul în care poate fi calculată valoarea D_1. Dacă stația centrală se află într-un punct p, atunci cel mai lung traseu care trece prin p este determinat de cele mai depărtate două stații față de p. Dacă aceste două stații se află în punctele q și r, atunci vom avea $D_1 =$

$d(p, q) + d(p, r)$, unde $d(x, y)$ reprezintă distanța dintre punctele x și y. Vom lua în considerare toate cele n posibilități de a alege punctul p și vom alege acel punct pentru care suma distanțelor de la el la cele mai depărtate două stații față de punctul respectiv este minimă. Cele mai depărtate două stații față de un punct pot fi determinate în timp liniar, deci, datorită faptului că vom efectua n astfel de determinări, ordinul de complexitate al acestei operații este $O(n^2)$.

Pentru a ilustra modul în care se poate calcula valoarea D_2 vom folosi un exemplu simplu. Trebuie observat faptul că, în acest caz, cel mai lung traseu dintre două stații va trece prin ambele stații centrale.

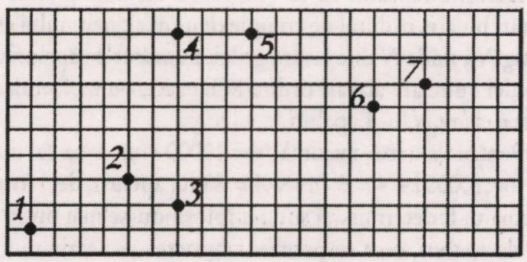


Figura 1

În figura 1 este prezentată o rețea cu șapte stații. Vom lua în considerare toate posibilitățile de a alege stațiile centrale și vom reține perechea pentru care diametrul rețelei este minim. Pentru început vom considera că avem $D_2 = \infty$. Pentru fiecare pereche de stații centrale (H_1 și H_2) vom considera că acestea sunt fixate. Celelalte n - 2 puncte (stații), vor fi inițial legate la una dintre cele două stații centrale (de exemplu la H_1).

Vom sorta aceste n - 2 puncte în ordine crescătoare în funcție de distanța față de H_1, rezultatul sortării fiind păstrat într-un vector P ($P[1]$ va fi cea mai apropiată stație față de H_1, iar $P[n - 2]$ va fi cea mai depărtată). Rezultatul sortării este ilustrat în figura 2.

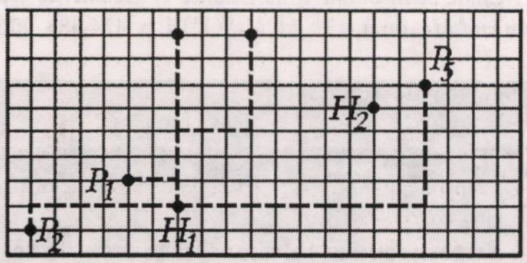


Figura 2

Vom nota prin r_1 valoarea $d(H_1, P[n - 3])$, prin r_2 valoarea $d(H_2, P[n - 2])$ și prin d_{12} valoarea $d(H_1, H_2)$. Dacă avem $r_1 + r_2 + d_{12} < D_2$, atunci punctul $P[n - 2]$ va fi legat de stația H_2 și valoarea D_2 va deveni $D_2 = r_1 + r_2 + d_{12}$. Acest pas este ilustrat în figura 3.

Avem $r_1 = d(H_1, P[n - 3]) = d(H_1, P[4]) = 10$, $r_2 = d(H_2, P[n - 2]) = d(H_2, P[5]) = 3$, $d_{12} = d(H_1, H_2) = 12$, deci $D_2 = r_1 + d_{12} + r_2 = 10 + 12 + 3 = 25$.

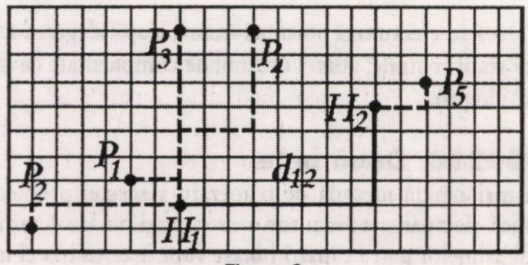


Figura 3

În continuare, aplicăm aceeași metodă pentru $r_1 = d(H_1, P[n-4])$, $r_2 = d(H_2, P[n - 3])$ și aceeași valoare $d_{12} = d(H_1, H_2)$ și obținem $r_1 + r_2 + d_{12} = d(H_1, P[3]) + d(H_1, H_2) + d(H_2, P[4]) = 7 + 12 + 8 = 27$. Deoarece noua distanță este mai mare decât diametrul anterior, valoarea D_2 rămâne nemodificată. Dacă, la final, valoarea D_2 va rămâne 25, atunci punctul $P[4]$ va fi legat de H_1 cu toate că distanța sa față de H_2 este mai mică decât distanța față de H_1. Situația este ilustrată în figura 4, în care punctul $P[4]$ este legat printr-o linie îngroșată de H_2 deoarece distanță față de H_1 este mai mare.

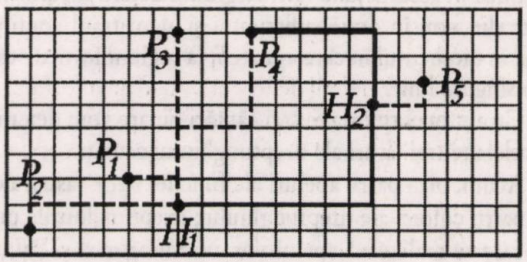


Figura 4

Algoritmul continuă luându-se în considerare celelalte elemente din P (în ordine descrescătoare a indicilor) până în momentul în care indicele devine 1. În figura 5 este ilustrată situația rețelei la final; valoarea D_2 a rămas 25.

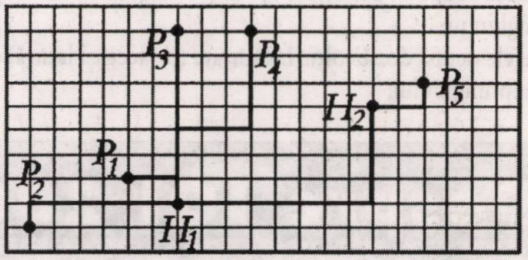


Figura 5

Pentru a obține ordinul de complexitate $O(n^3)$, trebuie respectată următoarea ordine de efectuare a operațiilor.

- Se iau în considerare toate punctele p ca fiind prima stație centrală.
- Se sortează celelalte n-1 puncte în funcție de distanța față de p.
- Se iau în considerare toate punctele q ca fiind a doua stație centrală.
- Se aplică algoritmul de determinare a diametrului pentru stațiile centrale p și q.

Dacă se efectuează sortarea numai după alegerea ambelor stații centrale, atunci ordinul de complexitate devine $O(n^3 \cdot \log n)$.

P060218: Două bețe

Cea mai rapidă metodă de rezolvare a acestei probleme implică efectuarea a cinci sau șase căutări binare. Cu ajutorul primelor patru căutări binare vom determina cel mai mic dreptunghi care conține în interiorul său ambele bețe.

Pentru prima căutare binară vom apela funcția `rect` pentru dreptunghiuri delimitate în stânga, dreapta și dedesubt de marginile grilei. Latura de sus a dreptunghiului va fi "variabilă" și, în urma căutării binare, va fi linia pe care se află cea mai de jos celulă care face parte dintr-un băț. Inițial, latura de sus va fi la jumătatea grilei. Dacă în urma apelului se returnează valoarea 0, atunci înseamnă că nu există în jumătatea de jos nici o celulă care face parte dintr-un băț și căutarea continuă în jumătatea de sus. Dacă se returnează valoarea 1, înseamnă că în jumătatea de jos există cel puțin o celulă care face parte dintr-un băț și căutarea continuă în această jumătate. Regiunea determinată se împarte din nou în două subregiuni și algoritmul continuă până în momentul în care ajungem la regiuni formate dintr-o singură linie.

Cu ajutorul celorlalte trei căutări binare vom determina celelalte trei laturi ale dreptunghiului căutat.

Acum, prin patru apeluri ale funcției `rect` (asupra celor patru colțuri ale dreptunghiului; dreptunghiurile pentru care se realizează apelul vor avea lungimea și lățimea egală cu unitatea, deci vor fi pătrate de latură 1) putem determina forma generală a structurii. În funcție de numărul și poziția colțurilor care conțin celule ocupate de bețe, deosebim patru cazuri (există și alte variante care pot fi obținute prin rotații):

- bețele ocupă trei dintre colțurile dreptunghiului;
- bețele ocupă două colțuri opuse (în diagonală) ale dreptunghiului;
- bețele ocupă două colțuri alăturate (pe aceeași latură) ale dreptunghiului;
- bețele nu ocupă nici unul dintre colțurile dreptunghiului.

Cele patru posibilități sunt ilustrate în figura 1 .

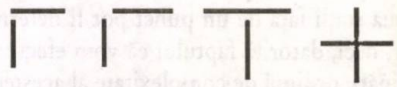


Figura 1

Pentru a determina poziția exactă a bețelor mai avem nevoie de una sau două căutări binare în funcție de forma generală a structurii.

Datorită faptului că se realizează un număr finit de căutări binare, ordinul de complexitate al algoritmului este $O(\log N)$, unde N este latura grilei; numărul de apeluri ale funcției `rect` are același ordin. Mai exact, vom efectua cel mult $6 \cdot \lceil log_2 N \rceil + 4$ apeluri.

Pentru valoarea maximă $N = 10000$ vom avea cel mult $6 \cdot \lceil \log_2 10000 \rceil + 4 = 6 \cdot 14 + 4 = 88$ de apeluri, deci întotdeauna va fi determinată soluția folosindu-se mai puțin de 100 de apeluri, ceea ce permite obținerea punctajului maxim.

Bibliografie

1. IOI 2002 Scientific Committee, *IOI 2002 Competition*,
2. A.B. Kahng, G. Robbins, *Optimal Algorithms Extracting Spatial Regularity in Images*, *Pattern Recognition Letters*, 12, p. 757-764, 1991
3. P. Brucker, *Efficient Algorithm for some Path Problems*, *Discrete Applied Mathematics*, 62, p. 77-85, 1995
4. S. Albers, P. Brucker, *The Complexity of One-machine Batching Problems*, *Discrete Applied Mathematics*, 47, p. 87-107, 1993
5. J. M. Ho, D. T. Lee, C. H. Chang, C. K. Wong, *Minimum Diameter Spanning Trees and Related Problems* , *SIAM J. on Computing*, 20(5), p. 987-997, 1991
6. T. Chan, *Semi-online Maintenance of Geometric Optima and Measures*, 13th *ACM-SODA*, p. 474-483, 2002

Despre IOI 2002...

La IOI 2002 concurenții au avut la dispoziție calculatoare dotate cu procesoare Intel Pentium 4 cu frecvența de 1,7 GHz și 256 MB de memorie RAM. În timpul concursului putea fi folosit fie sistemul de operare Microsoft Windows XP, fie Debian GNU/Linux 3.0.

Compilatoarele oficiale la IOI 2002 au fost GCC 2.95.3 și FPC 1.0.6. Mediile de programare puse la dispoziția concurenților au fost FreePascal IDE și RHIDE.

Au fost două zile de concurs în care participanții au avut la dispoziție cinci ore pentru a rezolva cele trei probleme propuse.

În ziua premergătoare începerii concursului propriu-zis a avut loc o sesiune de antrenament care avea ca scop familiarizarea concurenților cu modul în care se va desfășura concursul.

Mai multe detalii referitoare la IOI 2002 puteți găsi la adresa www.ioi2002.or.kr.

Rundele #01 - #04

Vom prezenta în cadrul acestui articol câteva informații referitoare la desfășurarea primelor patru runde ale concursului de programare organizat de revista noastră. De asemenea, veți putea vedea clasamentul primilor o sută clasați după desfășurarea acestor runde.

P070207: Găuri de vierme

Această problemă este o variantă simplă a problemei ***Sistem***, propusă spre rezolvare la ediția din acest an a *Olimpiadei Naționale de Informatică*.

Problema de la *ONI* 2002 cerea determinarea numărului grafurilor orientate cu N noduri în care fiecare nod are gradul 2.

Problema ***Găuri de vierme*** poate fi enunțată astfel: *să se determine numărul grafurilor orientate cu N noduri și fără bucle în care fiecare nod are gradul interior* 1 *și gradul exterior* 1.

O metodă simplă de rezolvare este asemănătoare celei prezentate pentru problema de la *ONI* 2002, diferențele fiind minime. Din nefericire, folosirea acestei metode duce la obținerea unui program care depășește limita de timp admisă.

Datorită faptului că intrarea consta într-un singur număr (cuprins între 2 și 255) putea fi folosită "metoda constantelor" pentru a obține un program cu un timp de execuție minim.

O altă modalitate de rezolvare este obținerea unei formule mai simple, care implică efectuarea unui număr mult mai mic de calcule. Aceasta putea fi obținută fie transformând formula de la prima metodă, fie "de la zero" observând faptul că problema este echivalentă cu cea a determinării numărului permutărilor σ cu N elemente care nu au puncte fixate ($\sigma_k \neq k$ pentru orice valoare k cuprinsă între 1 și N). Se ajunge la următoarea relație de recurență: $G_N = N \cdot G_{N-1} + (-1)^N$, unde $G_1 = 0$ și G_N reprezintă numărul grafurilor cu N vârfuri care respectă condițiile date.

Așa cum era de așteptat pentru problema care a marcat începutul concursului nostru, rezolvarea s-a dovedit a fi foarte ușor de găsit, 82 dintre participanții (care nu au fost descalificați ulterior) reușind să obțină punctajul maxim. Aceștia sunt: *Adrian Balașko (Zalău), Adrian Cârcu (Bistrița), Adrian-Marius Dumitran (București), Alexandra Constantin (Ploiești), Alexandru Constantin (Pitești), Alexandru Ivan (București), Alexandru Moșoi (Bacău), Alexandru Radu (București), Alexandru Sofronia (Bacău), Alin Coman (Târgu-Jiu), Alpár-Ferenc Perini (Sfântu Gheorghe), Andrei Benea (Focșani), Andrei Csibi (Cluj-Napoca), Andrei-Gabriel Ion (Constanța), Andrei Gheorghe (București), Andrei Homescu (Târgu-Jiu), Andrei Matei (București), Andrei Negoiță (Bacău), Angelo Lupuț (Satu Mare), Aurelian Marinescu (Brașov), Bogdan-Andrei Chirițoiu (Ploiești), Bogdan Nicolae (București), Bogdan Pănuș (Satu Mare), Călin Bindea (Bistrița), Călin Ciutu (București), Ciprian Cană (Vaslui), Ciprian-Nicolae Baicu (București), Ciprian-George Pop (Baia Mare), Constantin Jucovschi (Chișinău, Republica Moldova), Constantin Răuțu (Botoșani), Cornel Mihăilă (Brașov), Cosmin Raianu (Constanța), Cristian Alexandrescu (București), Csaba András (Oradea), Csaba Pătcaș (Cluj-Napoca), Dan Ghinea (București), Daniel Nechifor (Vaslui), Daniel Păsăilă (Suceava), Dan-Ionuț Fechete (Suceava), Dumitru Codreanu (Chișinău, Republica Moldova), Erdal Medayin Rasid (Tulcea), Eszter Tasi (Sfântu Gheorghe), Gawrychowski Pawel (Wroclaw, Polonia), Ioan-Claudiu Gruia (București), Ionuț Ene (Turturești), Iulian Blajut (Bacău), Ivan Kazmenko (Sankt Petersburg, Federația Rusă), Lucian Vancea (Blaj), Marian-Gheorghe Raica (Drăgușești), Marius Airinei (Bacău), Marius Nuca (Măgura), Marius-Dorin Moraru (Buftea), Maximilian Machedon (București), Mihail Popa (București), Mihail-Cosmin Pițt-Rada (Drobeta Turnu-Severin), Mircea Digulescu (București), Mircea-Șerban Savin (Iași), Miron Emilian (Bacău), Mugurel-Ionuț Andreica (București), Octavian-Daniel Dumitran (București), Radu Berinde (București), Radu Vatavu (Suceava), Răzvan Albu (București), Răzvan Dinu (Ploiești), Răzvan Tigaeru (Călărași), Silvestru-Cosmin Negrușeri (Bistrița), Silviu Hagea (Călărași), Slotta Johannes (Jena, Germania), Sorin Stancu-Mara (București), Sorin-Ionuț Balasioiu (Craiova), Ștefan Bucur (București), Ștefan Ciobâcă (Suceava), Ștefan Nour (Piatra Neamț), Valentin Bran (Satu Mare), Valentin Dinu (Pitești), Valentin Gușă (București), Victor Vernescu (Constanța), Victor-Marius Costan (București), Vincent Groenhuis (Enschede, Olanda) și Vlad Dascălu (Bacău).*

Pentru această problemă, 142 dintre concurenții înscriși au trimis soluții care au fost acceptate.

P070208: Matrice

Din modalitatea de acordare a punctajului rezulta destul de clar că această problemă este, de fapt, "un concurs de euristici". Nu este cunoscut până în prezent nici un algoritm polinomial de rezolvare a acestei probleme care să ducă la obținerea unui rezultat în care se folosește un număr minim de cvadrupli. Așadar, concurenții și-au folosit imaginația pentru a găsi o metodă de rezolvare care să ducă la obținerea de rezultate cât mai bune. Iată și lista primilor 20 clasați în această "competiție".

1. Adrian Cârcu, Bistrița	*98*
2. Dumitru Codreanu, Republica Moldova	*87*
3. Constantin Jucovschi, Republica Moldova	*79*
4. Angelo Lupuț, Satu Mare	*76*
5. Daniel Nechifor, Vaslui	*75*
6. Aurelian Marinescu, Brașov	*69*
6. Ciprian Cană, Vaslui	*69*
6. Ivan Kazmenko, Federația Rusă	*69*
6. Mugurel-Ionuț Andreica, București	*69*
10. Costel-Cătălin Antohi, Tecuci	*68*
11. Alexandru Constantin, Pitești	*66*
11. Cornel Mihăilă, Brașov	*66*
11. Dan-Ionuț Fechete, Suceava	*66*
11. Iulian Blajut, Bacău	*66*
11. Livian Disa, Băilești	*66*
11. Lucian Suditu, Vaslui	*66*
11. Marius Nicolae, Slatina	*66*
11. Mircea Digulescu, București	*66*
11. Tiberiu Chiorean, Baia Mare	*66*
11. Vlad Dascălu, Bacău	*66*

La această rundă au fost evaluate soluțiile trimise de către 126 dintre concurenți înscriși la această ediție.

P070209: Ecuații

Aceasta a fost prima problemă propusă spre rezolvare la concursul nostru în care fișierele de intrare erau cunoscute. Așadar, nu trebuia furnizat un program de rezolvare, ci doar fișierele de ieșire corecte. Majoritatea sistemelor de ecuații erau liniare, deci puteau fi rezolvate folosind metoda lui *Gauss* (programele ar fi trebuit să ruleze câteva ore). Pentru celelalte fișiere era necesară folosirea altor metode, dar soluțiile puteau fi găsite într-un interval rezonabil de timp.

Pentru această problemă, deși până la urmă s-a dovedit că am comis unele erori în procesul de evaluare (motiv pentru care ne cerem și acum scuze), au obținut punctajul maxim următorii concurenți: *Adrian Cârcu (Bistrița), Andrei Matei (București), Ciprian Cană (Vaslui), Cristian Bojinovici (București), Csaba Pătcaș (Cluj-Napoca), Dan Ghinea (București), Ivan Kazmenko (Sankt Petersburg, Federația Rusă), Mircea Digulescu (București), Mugurel Ionuț Andreica (București), Natalia Baranova (Sankt Petersburg, Federația Rusă) și Silvestru-Cosmin Negrușeri (Bistrița).*

Pentru această problemă doar 54 de concurenți au trimis fișiere de ieșire.

P070210: Șah

Se observă că numărul total al configurațiilor valide este destul de mic, deci pentru determinarea arborelui de joc poate fi folosită o tehnică bazată pe metoda *programării dinamice*. După construirea arborelui, pentru fiecare poziție validă se poate determina care este mutarea care duce la obținerea poziției de mat în număr minim de mutări, indiferent de mutările jucătorului negru.

Pentru a respecta limita de timp admisă trebuia construit un tablou de constante în care se păstra, pentru fiecare configurația validă, mutarea care trebuie efectuată, construirea proriu-zisă a arborelui de joc nefiind posibilă.

Au fost și concurenți care au reușit să obțină toate cele 100 de puncte folosind alte metode care nu implicau utilizarea unui șir de constante.

Problema s-a dovedit a fi destul de dificilă, doar șase dintre cele doar 41 de rezolvări trimise fiind punctate cu 100 de puncte. Concurenții care au primit punctajul maxim sunt:

Adrian Cârcu, Bistrița
Csaba András, Oradea
Ivan Kazmenko, Federația Rusă
Radu Ștefan, Brașov
Silvestru-Cosmin Negrușeri, Bistrița
Victor-Marius Costan, București

Doar cinci alți concurenți au reușit să obțină peste 50 de puncte la această problemă:

Dumitru Codreanu, Republica Moldova	*94*
Mircea Digulescu, București	*81*
Ionuț-Bogdan Petre, Plopeni	*78*
Alin Coman, Târgu-Jiu	*55*
Tiberiu Chiorean, Baia Mare	*50*

Despre concurs

La primele patru runde au fost descalificați mai mulți concurenți datorită faptului că au trimis soluții identice și aproape identice.

De asemenea, mai mulți participanți nu au primit punctele așteptate deoarece arhivele pe care le-am primit erau corupte. Vă rugăm să vă asigurați că arhivele pe care le trimiteți sunt transferate corect, adăugând propria dumneavoastră adresă de *e-mail* în câmpul CC al mesajului pe care îl trimiteți. Evident, nu este 100% sigur că, dacă arhiva care se întoarce la dumneavostră este intactă, atunci și cea pe care o primim noi este, dar dacă observați că "primiți" o arhivă invalidă, atunci puteți fi aproape siguri că trebuie să o retrimiteți. *Vom accepta întotdeauna cea de-a doua arhivă dacă prima pe care o primim este coruptă!*

Pe pagina alăturată este prezentată lista primilor 100 clasați după primele patru etape ale concursului de programare **Bursele Agora**. *Mulțumim tuturor concurenților pentru participarea la concursul nostru!!!*

Clasament după 4 runde

1. Adrian Cârcu, Bistrița **398**
2. Ivan Kazmenko, Sankt Petersburg - Federația Rusă **369**
3. Silvestru-Cosmin Negruțeri, Bistrița **364**
4. Dumitru Codreanu, Chișinău - Republica Moldova **361**
5. Mircea Digulescu, București **347**
6. Ciprian Cană, Vaslui **317**
6. Mugurel-Ionuț Andreica, București **317**
8. Dan Ghinea, București **302**
9. Aurelian Marinescu, Brașov **299**
10. Radu Vatavu, Suceava **274**

11. Andrei Matei, București 269
12. Victor Costan, București 264
13. Cornel Mihăilă, Brașov 256
14. C. Jucovschi, Rep. Moldova 252
15. Marian Raica, Drăgușești 248
16. Ștefan Ciobâcă, Suceava 245
17. Bogdan Nicolae, București 243
18. Andrei Homescu, Târgu-Jiu 231
19. Radu Berinde, București 224
20. Vlad Dascălu, Bacău 218
21. Angelo Lupuț, Satu Mare 217
22. Călin Ciutu, București 214
23. Csaba András, Oradea 200
23. Csaba Pătcaș, Cluj-Napoca 200
25. Alex. Constantin, Pitești 196
25. Iulian Blajut, Bacău 196
27. Daniel Nechifor, Vaslui 195
28. Tiberiu Chiorean, Baia Mare 190
29. Ionuț-Bogdan Petre, Plopeni 188
30. Dan-Ionuț Fechete, Suceava 182
31. Marius Airinei, Bacău 176
32. Răzvan Albu, București 175
33. Ștefăniță Fechete, Suceava 166
34. Alex. Constantin, Ploiești 164
34. M. Pit-Rada, Dr. Tr. Severin 164
36. Alexandru Sofronia, Bacău 163
37. Marius Moraru, Buftea 162
38. Alexandru Moșoi, Bacău 160
38. Andrei Ion, Constanța 160
40. Călin Bindea, Bistrița 159
41. Alin Coman, Târgu-Jiu 155
41. Saratov Team, Rusia 155
41. Ștefan Bucur , București 155
44. Natalia Baranova, Rusia, 154
45. Valentin Bran, Satu Mare 151
46. Bogdan Chiriținoiu, Ploiești 143
47. Bogdan Pănuș, Satu Mare 141
48. Daniel Păsăilă, Suceava 138
49. Vincent Groenhuis, Olanda 137
50. Silviu Hagea, Călărași 135
51. Mircea Pasoi, Ploiești 134
52. Mihail Popa, București 128
53. Cristian Bojinovici, București 122
54. M. Machedon, București 120
55. Miron Emilian, Bacău 120
56. Iuliu Konya, Baia Mare 116
57. Ștefan Nour, Piatra Neamț 114
58. Ciprian Baicu, București 109
59. Alpár Perini, Sf. Gheorghe 108
60. Lucian Suditu, Vaslui 108
61. Mircea-Șerban Savin, Iași 106
62. Cosmin Raianu, Constanța 104
63. Ionuț Ene, Turturești 104
64. Victor Vernescu, Constanța 104
65. Laura Drăgoi, Prahova 101
66. Adrian Balașko, Zalău 100
66. Marius Dumitran, București 100
66. Al. Sande, Lindiceni, 100
66. Alexandru Ivan, București 100
66. Alexandru Radu, București 100
66. Andrei Benea, Focșani 100
66. Andrei Csibi, Cluj-Napoca 100
66. Andrei Gheorghe, București 100
66. Andrei Negoiță, Bacău 100
66. Ciprian Pop, Baia Mare 100
66. Constantin Răuțu, Botoșani 100
66. C. Alexandrescu, București 100
66. Erdal Medayin Rasid, Tulcea 100
66. Eszter Tasi, Sf. Gheorghe 100
66. G. Pawel, Wroclaw 100
66. Claudiu Gruia, București 100
66. Lucian Vancea, Blaj 100
66. Marius Nuca, Măgura 100
66. Daniel Dumitran, București 100
66. Radu Ștefan, Brașov 100
66. Răzvan Dinu, Ploiești 100
66. Răzvan Tigaeru, Călărași 100
66. Slotta Johannes, Germania 100
66. S. Stancu-Mara, București 100
66. Ionuț Balasioiu, Craiova 100
66. Valentin Dinu, Pitești 100
66. Valentin Gușă, București 100
93. Marius Nicolae, Slatina 87
94. Cătălin Antohi, Tecuci 80
95. Mihai Cosma, Bistrița 76
96. Alex. Tomescu, București 67
96. G. Matei, Câmpulung Muscel 67
98. Livian Disa, Băilești 66
99. Andrei-Eugeniu Ioniță, Slatina 64
99. A. Paponiu, Dr. Tr. Severin 64

Bursele Agora

Vă prezentăm în continuare enunțurile problemelor propuse spre rezolvare la etapele #05-#08 ale ediției 2002/2003 a concursului de programare Bursele Agora.

P080215: Perechi

Se consideră un număr natural n și trebuie formate cât mai multe perechi din care trebuie să facă parte numere naturale cuprinse între 1 și $2 \cdot n$ astfel încât suma pătratelor celor două numere din oricare dintre perechi să fie număr prim. Fiecare număr poate face parte din cel mult o pereche.

Date de intrare

Fișierul de intrare **PAIRS.IN** conține o singură linie pe care se află valoarea n.

Date de ieșire

Prima linie a fișierului de ieșire **PAIRS.OUT** va conține numărul k al perechilor formate. Fiecare dintre următoarele k linii va conține câte două numere cuprinse între 1 și $2 \cdot n$, astfel încât suma pătratelor celor două numere este număr prim. Perechile pot fi scrise în orice ordine.

Restricție

- $1 \le n \le 2500$.

Exemplu

PAIRS.IN

```
7
```

PAIRS.OUT

```
7
1 4
2 3
5 6
7 8
9 10
11 14
12 13
```

Modalitatea de acordare a punctajului

Dacă pentru un test se pot obține cel mult X puncte, punctajul acordat va fi $X / 2^{n-k}$, unde n este numărul din fișierul de intrare, iar k este numărul perechilor formate. Valoarea va fi rotunjită la două zecimale exacte. Dacă nu se obține nici o pereche sau există erori în fișierul de ieșire nu se va acorda nici un punct.

Timp de execuție: 1 secundă/test

P080216: Numere prime în 3D

Se consideră un tablou tridimensional A de dimensiune $M \times N \times P$ ale cărui elemente trebuie completate astfel încât orice element are valoarea egală cu numărul elementelor vecine care sunt prime.

Două elemente A_{ijk} și $A_{i'j'k'}$ sunt vecine dacă și numai dacă $\max(|i - i'|, |j - j'|, |k - k'|) = 1$, $1 \le i, i' \le M$, $1 \le j, j' \le N$, $1 \le k, k' \le P$.

Valoarea **0** poate fi folosită ca *joker*, în sensul că unui element i se poate atribui această valoare indiferent de numărul elementelor vecine care sunt prime.

Punctajul unui tablou este dat de suma elementelor sale.

Va trebui să completați tabloul astfel încât punctajul său să fie cât mai mare posibil.

Date de intrare

Fișierul de intrare **3D.IN** conține valorile M, N și P, separate prin câte un spațiu.

Date de ieșire

Fișierul de ieșire **3D.OUT** va avea $M \times N \times P$ linii. Fiecare trebuie să conțină câte un număr reprezentând valoarea unui element al tabloului. Elementele tabloului vor fi ordonate crescător în funcție de primul index i, pentru valori egale ale primului index în funcție de al doilea index j, iar pentru valori egale ale primilor doi indecși în funcție de al treilea index k. De exemplu, pentru $M = N = P = 2$, ordinea va fi A_{111}, A_{112}, A_{121}, A_{122}, A_{211}, A_{212}, A_{221}, A_{222}.

Restricții

- $1 \le M, N, P \le 30$.

Exemplu

3D.IN

```
2 2 2
```

3D.OUT

```
7
7
7
7
7
7
7
7
```

Modalitatea de acordare a punctajului

Vom considera că, pentru fiecare test, se vor putea obține cel mult X puncte.

Concurenții care vor obține cea mai mare valoare *PtMax* pentru punctajul tabloului vor primi X puncte pentru testul respectiv.

Ceilalți concurenți, care au completat corect tabloul A și au obținut un punctaj Pt, vor obține $X \cdot Pt / PtMax$ puncte pentru testul respectiv. Această valoare va fi aproximată cu două zecimale exacte.

Punctajul final va fi obținut prin adunarea punctajelor de la fiecare test și rotunjirea acestuia la cel mai apropiat număr întreg.

Dacă tabloul nu este completat corect, concurenții nu vor primi nici un punct pentru testul respectiv.

De exemplu, dacă pentru un test se pot obține cel mult 5 puncte, cel mai bun rezultat obținut de un concurent constă într-un tablou cu punctajul 56, iar un alt concurent obține o soluție cu punctajul 25, atunci, pentru testul respectiv, punctajul concurentului va fi de $5 \cdot 25 / 56 = 2.23$ puncte.

Timp de execuție: 1 secundă/test

P080217: Factorizare

Se consideră un număr natural N și trebuie determinată o pereche de numere naturale strict mai mari decât 1, al căror produs să fie egal cu N.

Date de intrare

La această problemă fișierele de intrare sunt deschise. În pagina **http://www.ginfo.ro/concurs/runda07/index.shtml** sunt disponibile *link*-uri spre douăzeci de fișiere de intrare.

Numele acestora au forma **FACTORXX.IN**, unde **XX** ia valori între **01** și **20** și reprezintă numărul testului. Fiecare dintre cele 20 de fișiere de intrare conține o singură linie pe care se află valoarea N.

Date de ieșire

Pentru această problemă nu va trebui să trimiteți un program care să o rezolve, ci doar cele 20 de fișiere de ieșire corecte. Acestea vor fi denumite **FACTORXX.OUT**, unde **XX** ia valori între **01** și **20** și reprezintă numărul testului.

Prima linie a fișierului de ieșire va conține textul `FACTORS - TEST #XX`, unde `XX` reprezintă numărul testului.

Următoarele două linii vor conține câte un număr strict mai mare decât 1. Produsul acestor două numere trebuie să fie egal cu numărul N din fișierul de intrare corespunzător.

Dacă nu există două astfel de numere înseamnă că valoarea N este un număr prim, iar fișierul de ieșire va conține doar două linii; pe prima se va afla textul `FACTORS - TEST #XX`, unde `XX` reprezintă numărul testului, iar pe a doua se va afla textul `PRIME NUMBER`.

Exemplu

Vom considera că acest exemplu reprezintă testul **00**.

FACTOR00.IN

```
221
```

FACTOR00.OUT

```
FACTORS - TEST #00
13
17
```

Modalitatea de acordare a punctajului

Pentru fiecare fișier de ieșire corect veți obține un anumit număr de puncte. Punctajul maxim care poate fi obținut (dacă toate cele 20 de fișiere de ieșire sunt corecte) este de 100 de punte.

Pentru primele patru teste (**01 - 04**) se vor acorda câte 3 puncte, pentru următoarele patru (**05 - 08**) câte 4 puncte, pentru următoarele patru (**09 - 12**) câte 5 puncte, pentru următoarele patru (**13 - 16**) câte 6 puncte, iar pentru ultimele patru (**17 - 20**) câte 7 puncte.

Trimiterea fișierelor de ieșire

Arhiva pe care o veți trimite se va numi **YYYYYR07.ZIP**, **YYYYYR07.RAR** sau **YYYYYR07.ACE** (în funcție de programul de arhivare pe care îl folosiți), unde **YYYYY** este codul dumneavoastră de identificare (ID).

Arhiva nu va conține fișiere batch (**YYYYYR07.BAT**) sau fișiere sursă (**YYYYYR07.PAS**, **YYYYYR07.CPP** sau **YYYYYR07.C**). În arhivă se vor afla cele 20 de fișiere de ieșire corecte, denumirea lor fiind **FACTORXX.OUT**, unde **XX** reprezintă numărul testului (cuprins între **01** și **20**).

P080218: Graf

Se consideră un graf neorientat cu N noduri și N muchii în care fiecare nod are exact ***două*** muchii adiacente (fiecare nod are gradul 2).

Costurile muchiilor sunt numere prime, dar sunt necunoscute. Costul unui nod este dat de produsul costurilor muchiilor adiacente nodului respectiv. Nodurile sunt identificate prin numere cuprinse între 1 și N, iar costurile lor sunt cunoscute.

Va trebui să determinați costurile muchiilor folosind datele furnizate de o bibliotecă externă. Biblioteca vă va pune la dipoziție rutine pentru:

- inițializare;
- determinarea numărului de noduri ale grafului;

- determinarea costului unui nod;
- verificarea existenței unei muchii între două noduri;
- verificarea costului unei muchii;
- furnizarea ca rezultat a costului unei muchii;
- determinarea numărului maxim de verificări ale costurilor muchiilor.

Programul vostru va trebui să determine costurile muchiilor folosind un număr limitat de apeluri ale funcției de verificare a costului unei muchii. Programul va trebui să furnizeze rezultate pentru toate cele N muchii ale grafului, deci vor exista N apeluri ale rutinei de furnizare a costului unui muchii.

Bibliotecile Pascal și C/C++

Pentru a putea folosi biblioteca va trebui să includeți în programul dumneavoastră istrucțiunea **uses** `graph`; pentru limbajul *Pascal* și `#include "graph.h"` pentru limbajul *C/C++*. În continuare sunt prezentate sintaxele funcțiilor și procedurilor incluse în biblioteci:

procedure `Init;`
void `Init()`
- trebuie apelată la începutul programului și este folosită de către bibliotecă pentru inițializare;
- întrerupe execuția programului dacă este apelată a doua oară.

function `GetNumberOfNodes:Integer;`
int `GetNumberOfNodes()`
- returnează numărul nodurilor (și al muchiilor) din graf.

function `IsEdgeBetween(x,y:Integer):Boolean;`
int `IsEdgeBetween(int x,int y)`
- returnează `true` (valoare nenulă) dacă există o muchie între nodurile `x` și `y` și `false` (0) dacă nu există o astfel de muchie;
- întrerupe execuția programului dacă cel puțin una dintre valorile `x` și `y` nu reprezintă un nod al grafului.

function `CheckEdgeWeight(x,y,c:Integer):Boolean;`
int `CheckEdgeWeight(int x,int y,int c)`
- returnează `true` (valoare nenulă) dacă muchia dintre nodurile `x` și `y` are costul `c` și `false` (0) dacă muchia are costul diferit de `c`;
- întrerupe execuția programului dacă cel puțin una dintre valorile `x` și `y` nu reprezintă un nod al grafului, dacă muchia determinată de aceste noduri nu face parte din graf sau dacă a fost depășit numărul maxim admis de apeluri ale acestei funcții.

function `GetNodeWeight(x:Integer):Longint;`
long `GetNodeWeight(int x)`
- returnează costul nodului x;
- întrerupe execuția programului dacă `x` nu reprezintă un nod al grafului.

function `GetNumberOfCallsLeft:Integer;`
int `GetNumberOfCallsLeft()`
- returnează un număr întreg care reprezintă numărul de apeluri ale funcției `CheckEdgeWeight` pe care le mai aveți la dispoziție.

function `Result(x,y,c:Integer):Boolean;`
int `Result(int x,int y,int c)`
- acceptă ca rezultat faptul că muchia dintre nodurile `x` și `y` are costul `c`;
- întrerupe execuția programului dacă:
  - cel puțin una dintre valorile `x` și `y` nu reprezintă un nod al grafului;
  - muchia determinată de aceste noduri nu face parte din graf;
  - costul muchiei dintre nodurile `x` și `y` nu este `c`;
  - costul muchiei dintre nodurile `x` și `y` este furnizat a doua oară;
  - au fost furnizate costurile tuturor celor N muchii ale grafului (*Acesta este singurul caz în care sunt acordate puncte!*).

Toate funcțiile și procedurile (excepție: `Init`) întrerup execuția programului dacă sunt apelate înaintea inițializării.

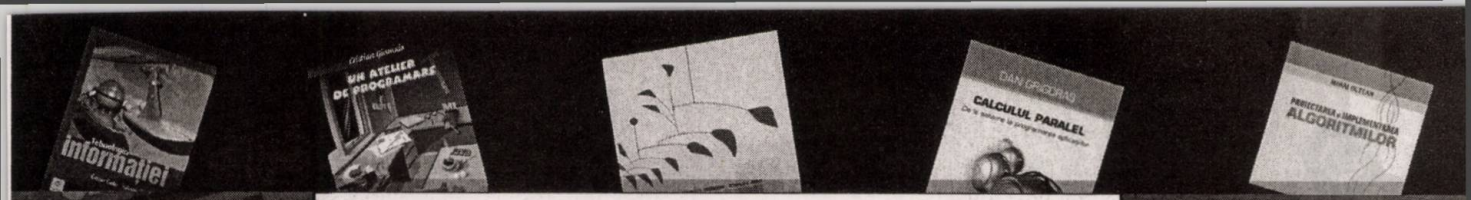
Restricții și precizări

- Programul vostru nu va citi date din nici un fișier de intrare și nu va scrie date în nici un fișier de ieșire.
- $3 \le N \le 1000$.
- Costurile muchiilor sunt numere prime cuprinse între 2 și 29989.

Exemplu de utilizare a bibliotecii

Subprogram apelat	*Valoare returnată*
`Init`	-
`GetNumberOfNodes`	4
`IsEdgeBetween(1,2)`	true (1)
`IsEdgeBetween(1,3)`	false (0)
`IsEdgeBetween(1,4)`	true (1)
`IsEdgeBetween(2,3)`	true (1)
`IsEdgeBetween(2,4)`	false (0)
`IsEdgeBetween(3,4)`	true (1)
`GetNumberOfCallsLeft`	2
`GetNodeWeight(1)`	14
`GetNodeWeight(2)`	6
`GetNodeWeight(3)`	15
`GetNodeWeight(4)`	35
`CheckEdgeWeight(1,2,7)`	false (0)
`CheckEdgeWeight(1,2,2)`	true (1)
`Result(1,2,2)`	-
`Result(2,3,3)`	-
`Result(3,4,5)`	-
`Result(4,1,7)`	-

Timp de execuție: 5 secunde/test



COMPUTER LIBRIS AGORA

Cartea este o culegere de studii de caz, construite pe temelia problemelor concursurilor Southeastern European Regional Programming Contest din perioada 1995-1999, concursuri organizate sub egida Association for Computing Machinery.

Cartea se adresează atât începătorilor, cât și celor cu experiență în domeniul programării, în încercarea de a contura prin exemple imaginea unei lumi diverse și pasionante, dar pretențioase, aceea a înzestrării mașinii de calcul cu capacitatea de a rezolva corect și eficient.

COMPUTER LIBRIS AGORA

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

Circuite BITONICE

Lucian Ilea, Vlad Țira

Problema determinării unui circuit bitonic a apărut, în diverse forme, la mai multe concursuri de programare. În acest articol vom prezenta o modalitate de rezolvare a acestei probleme.

În cele ce urmează va fi rezolvată următoarea problemă: "fiind cunoscute coordonatele în plan pentru n puncte diferite între ele, $P_i(x_i, y_i)$, $i = 1, \ldots, n$, să se determine un traseu circular de lungime totală minimă, traseu care trece exact o dată prin fiecare punct". Problema este propusă în [1] la pagina 279 și a fost întâlnită, sub diverse forme, la numeroase concursuri de informatică.

În cazul general problema este *NP - completă*. Există, însă, un algoritm cu timp de execuție $O(n^2)$, aplicabil dacă se presupune, suplimentar, că nu există puncte cu aceeași abscisă și că se caută doar circuitele bitonice, adică acelea care încep din punctul cel mai din stânga, merg apoi numai de la stânga la dreapta către punctul cel mai din dreapta, iar apoi merg numai către stânga, înapoi spre punctul de pornire. Situația este descrisă în desenele alăturate. În prima figură este reprezentat cel mai scurt circuit, de lungime 24,88..., iar în a doua figură este reprezentat cel mai scurt circuit bitonic, de lungime 25,58... Exemplul este preluat din [1], pagina 279.

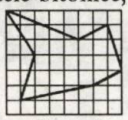


Figura 1

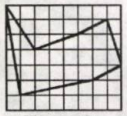


Figura 2

Pentru rezolvare se va folosi metoda *programării dinamice* și se va construi o matrice L, care are n linii și $n + 1$ coloane. Punctele se consideră ordonate crescător după abscise. Dificultățile legate de tratarea circuitelor vor fi soluționate prin introducerea unui punct suplimentar P_{n+1}, care este situat la o distanță foarte mică, în dreapta punctului P_n. Procedând în acest mod, în circuit se vor evidenția două părți, cea de la P_1 la P_n și cea de la P_1 la P_{n+1}, în ambele neexistând întoarceri. Introducem următoarele notații:

- $l_{i,j}$ - lungimea segmentului P_iP_j;
- $d_{i,j}$ - lungimea drumului "direct" între P_i și P_j, $d_{i,j} = l_{i,i+1} + l_{i+1,i+2} + \ldots + l_{j-1,j}$;
- $L_{i,j}$ $(i < j)$ lungimea celui mai scurt traseu fără întoarceri care are drept capete punctele P_i și P_j, care trece prin punctul P_1 și care conține toate punctele $P_1, P_2, \ldots, P_j$ (vezi figura 3).

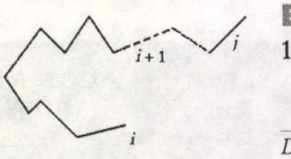


Figura 3

Rezultatul (lungimea celui mai scurt circuit bitonic) se va găsi în $L_{n,n+1}$. Modul de determinare al elementelor matricei L este următorul:

(1) $L_{i,j} = L_{i,i+1} + d_{i,j}$; dacă $i < j$;
(2) $L_{i,i+1} = \min\{k = 1, \ldots, i - 1 \mid L_{k,i} + l_{k,i+1}\}$.

Justificarea acestor formule este imediată, ținând cont de semnificația elementelor care intră în calcule. Matricea L va fi construită începând cu linia 1 și va conține 0 în pozițiile (i, j) pentru care $i \geq j$.

Pentru situația redată în figura 2 avem: $n = 7$, $P_1(0, 6)$, $P_2(1, 0)$, $P_3(2, 3)$, $P_4(5, 4)$, $P_5(6, 1)$, $P_6(7, 5)$, $P_7(8, 2)$ (punctele sunt ordonate crescător după abscise). Se introduce punctul $P_8(8.0001, 2)$. Lungimea celui mai scurt circuit bitonic este 25,58 (valoarea este rotunjită la două zecimale exacte). Cu excepția elementelor de pe poziții vecine cu diagonala principală a matricei L, restul elementelor se determină printr-o adunare, ceea ce demonstrează complexitatea $O(n^2)$ a acestui algoritm. Reconstituirea traseului în funcție de rezultatele din matrice se bazează, în esență, pe găsirea acelui k pentru care are loc egalitatea în formula (2).

Situații interesante pot surveni în cazul unor dispuneri particulare ale punctelor P_i. În figura 4 este reprezentat cazul în care acestea se obțin ca puncte de intersecție între două seturi de drepte paralele echidistante.

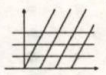


Figura 4

Considerăm că distanța dintre cele n drepte paralele cu axa Ox este d_1 și că distanța dintre cele n drepte paralele oblice este d_2. În total se obțin n^2 puncte de intersecție. Notând cu α măsura unghiului format de dreptele oblice cu axa Ox, o condiție suficientă să nu existe puncte de intersecție cu aceeași abscisă este ca $d_1/d_2 \cdot \cos(\alpha)$ să nu fie număr rațional. În cazul în care $n = 6$, $d_1 = d_2 = 10$, $\alpha = \pi/4$, se obține că lungimea celui mai scurt circuit bitonic este 509,61, iar traseul este cel reprezentat în figura 5.

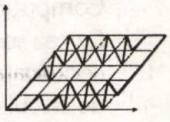
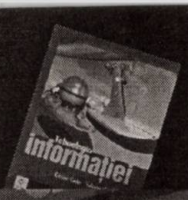
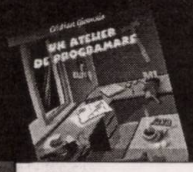


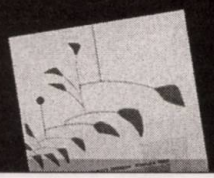
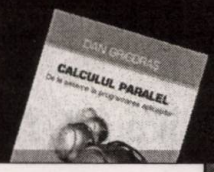
Figura 5

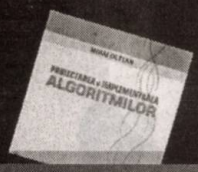
Bibliografie

1. T. Cormen, C. Leiserson, R. Rivest, *Introducere în algoritmi*, Computer Libris Agora, Cluj-Napoca, 2001

Dl. prof. Lucian Ilea este cadru didactic la Liceul "Emil Racoviță" din Cluj și poate fi contactat prin e-mail la adresa **ilucian@email.ro**. *Vlad Țira este elev în clasa a XI-a la Liceul "Emil Racoviță" din Cluj și poate fi contactat prin e-mail la adresa* **vldut@email.ro**.



COMPUTER LIBRIS AGORA

DAN GRIGORAȘ

CALCULUL PARALEL

De la sisteme la programarea aplicațiilor

Calculul paralel. De la sisteme la aplicații tratează, într-un mod natural și structurat, cele mai importante aspecte ale proiectării aplicațiilor paralele, sau, cu un termen mai nou, de mare performanță. Paralelismul, care înseamnă executarea mai multor operații simultan, este o caracteristică a tuturor calculatoarelor actuale, inclusiv a celor personale. De aceea, cunoașterea principiilor și legilor care îl guvernează, ca și a metodologiilor pentru proiectarea aplicațiilor cele mai performante este o necesitate,dacă se dorește exploatarea optimă a resurselor sistemelor de calcul. Parametrii de performanță și metodologiile de proiectare și execuție optimă reprezintă subiectele care guvernează discursul științific al cărții de față.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

Pagini WEB cu PHP 4

Mihai Scorțaru, Claudiu Soroiu

Reluăm seria de articole destinate proiectării paginilor Web cu descrierea unuia dintre cele mai cunoscute interpretoare folosite pentru a crea site-uri interactive. Începând cu acest număr vom prezenta cea de-a patra versiune a interpretorului PHP (Hypertext PreProcessor).

Introducere

Interpretorul *PHP* este cel mai cunoscut limbaj de *scripting* folosit în acest moment pentru crearea *site*-urilor *Web* interactive. Denumirea este un "acronim recursiv" pentru *Hypertext PreProcessor*.

Diferența esențială față de alte limbaje de *scripting* pe care le-am prezentat în articolele anterioare (de exemplu, *JavaScript*) este faptul că *PHP* este un interpretor *server-side* (operațiile sunt executate de către server și nu pe calculatorul utilizatorului). Pentru a putea testa pagini *PHP* aveți nevoie de un *server* de *Web* (de exemplu, *Abyss Web Server X1*) și de pachetul *PHP* instalat. Nu ar trebui să aveți probleme cu instalarea și configurarea acestor produse.

Primul script PHP

Ca de obicei vom începe cu cea mai simplă pagină *Web*, cea care afișează mesajul *Hello World!*

```
<HTML>
  <HEAD>
    <TITLE>
      Hello World!
    </TITLE>
  </HEAD>
  <BODY>
    <?php echo "Hello World!"; ?>
  </BODY>
</HTML>
```

Efectul acestui *script* este generarea unei pagini *Web* (pe *server*) cu următorul conținut:

```
<HTML>
  <HEAD>
    <TITLE>
      Hello World!
    </TITLE>
  </HEAD>
  <BODY>
    Hello World!
  </BODY>
</HTML>
```

Aceasta este clasica pagină de afișare a unui simplu mesaj.

Deocamdată nu se observă nici o diferență între codul *PHP* și un document *HTML* clasic, cele două pagini *Web* generate fiind identice indiferent de programul de navigare folosit pentru vizualizarea lor.

Primul script PHP "adevărat"

Vom prezenta acum un *script* care generează o pagină *Web* în care va fi afișat identificatorul programului de navigare folosit pentru vizualizarea ei.

Evident, conținutul paginii va fi diferit în funcție de *browser*-ul utilizat. Documentul *PHP* este prezentat în continuare:

```
<HTML>
  <HEAD>
    <TITLE>
      Browser
    </TITLE>
  </HEAD>
  <BODY>
    <H2>
    <?php
    echo $_SERVER["HTTP_USER_AGENT"];
  ?>
    </H2>
  </BODY>
</HTML>
```

Dacă folosim pentru vizualizare programul de navigare *Internet Explorer* 5.0 și sistemul de operare *Windows* '98, pagina *Web* generată de *server* este următoarea:

```
<HTML>
  <HEAD>
    <TITLE>
      Browser
    </TITLE>
  </HEAD>
  <BODY>
    <H2>
      Mozilla/4.0 (compatible; MSIE 5.0;
      Windows 98; DigExt)
    </H2>
  </BODY>
</HTML>
```

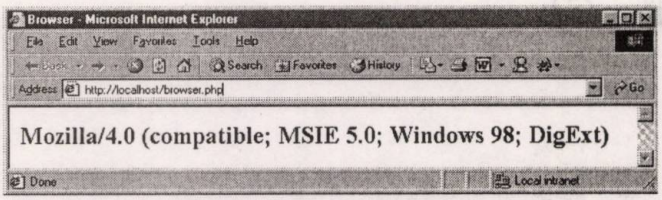


Figura 1: Internet Explorer 5.0

În figura 1 este prezentată o captură a ferestrei programului de navigare *Internet Explorer* 5.0 în momentul în care este afișată această pagină *Web*.

Dacă accesăm documentul *PHP* folosind același sistem de operare și programul de navigare *Opera* 7.0, va fi generată următoarea pagină *Web*:

```
<HTML>
  <HEAD>
    <TITLE>
      Browser
    </TITLE>
  </HEAD>
  <BODY>
    <H2>
      Opera/7.0 (Windows 98; U) [en]
    </H2>
  </BODY>
</HTML>
```

Pagina afișată de programul de navigare *Opera* 7.0 este prezentată în figura 2.

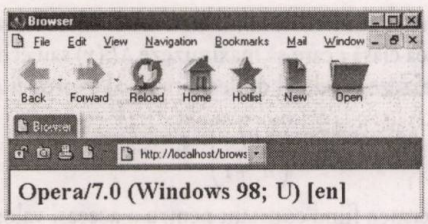


Figura 2: Opera 7.0

Un exemplu mai complex

Vom prezenta acum un script *PHP* care afișează data curentă a serverului. Codul este următorul:

```
<HTML>
  <HEAD>
    <TITLE>
      Data
    </TITLE>
  </HEAD>
  <BODY>
    <H2>
    <?php
      $luni=array(1
      => 'Ianuarie',
      'Februarie',
      'Martie',
      'Aprilie',
      'Mai','Iunie','Iulie','August',
      'Septembrie','Octombrie',
      'Noiembrie','Decembrie');
      $today = getdate();
      $year = $today['year'];
      $mon = $today['mon'];
      $mday = $today['mday'];
      echo "$mday $luni[$mon] $year";
    ?>
    </H2>
  </BODY>
</HTML>
```

Dacă documentul *PHP* a fost accesat pe data de 2 Decembrie 2002 (data *server*-ului), atunci pagina *Web* generată este cea din figura 3.

Figura 3: Data server-ului

Trebuie menționat faptul că folosind limbajul *JavaScript* nu există nici o posibilitate de a afla data curentă a *server*-ului, putând fi afișată doar data curentă de pe calculatorul celui care accesează pagina *Web*.

Despre PHP

PHP permite folosirea unor elemente specifice limbajelor de programare. Ieșirea standard a *script*-ului *PHP* devine intrare standard pentru programul de navigare care vizualizează pagina. Așadar, la ieșirea standard poate fi scris (de exemplu, prin intermediul comenzii **echo**) orice tip de cod *HTML*, acesta fiind interpretat corect de către *browser*.

În numărul următor...

Articolul următor dedicat interpretorului *PHP* va conține o scurtă descriere a sintaxei limbajului. Vor fi prezentate detalii referitoare la utilizarea variabilelor, constantelor, expresiilor, operatorilor, diferitelor structuri de control etc.

Mihai Scorțaru este redactor șef GInfo și poate fi contactat prin e-mail la adresa **skortzy@yahoo.com**. *Claudiu Soroiu este redactor GInfo și poate fi contactat prin e-mail la adresa* **csoroiu@yahoo.com**.

Serial

RECUNOAȘTEREA formelor

Radu-Daniel Vatavu

Serialul dedicat recunoșterii formulor se încheie cu prezentarea unor procedee de învățare nesupracegheată. Vor fi descrise diferite modalități de abordare cum ar fi metoda celor mai mici pătrate, metoda nucleelor dinamice, metode de clasificare ierarhice etc. De asemenea, vom realiza o prezentare comparativă a clasificatorilor descriși pe parcursul celor trei episoade ale serialului.

Învățarea nesupravegheată

Învățarea nesupravegheată are drept scop împărțirea unei mulțimi de forme într-un anumit număr de clase, în funcție de gradul de "asemănare" sau de similaritate dintre acestea. Spre deosebire de contextul supravegheat al recunoașterii formelor, caracterizat de prezența unui "profesor" sau "supraveghetor" astfel încât apartenența formelor din setul de învățare la clase este cunoscută, în cadrul contextului nesupravegheat singurele informații de care se dispune sunt legate doar de măsurătorile efectuate asupra formelor. Ca urmare, fie F mulțimea formelor de care se dispune și care trebuie clasificate într-un număr de m clase:

$$F = \{x_0, x_1, \ldots x_{p-1}\}.$$

Pentru început, vom considera numărul m de clase în care vor fi împărțite formele cunoscute.

Deci, problema constă în partiționarea mulțimii F în m clase cât mai omogene posibil.

$$F = \{F_0, F_1, \ldots, F_{m-1}\}$$
$$F_i \cap F_j = 0 \; \forall i, j \in \{0, 1, \ldots, m-1\}$$
$$i \neq j, F = \bigcup_{j=0}^{m-1} F_j.$$

Metoda celor mai mici pătrate

O primă abordare de rezolvare a problemei are drept punct de plecare metoda celor mai mici pătrate. În acest context, gruparea formelor în cele m clase se va face urmărindu-se minimizarea unui criteriu E de tipul:

$$E = \sum_{k=0}^{m-1} E_k, E_k = \sum_{x \in F_k} d^2(x, m_k),$$

unde d este distanța euclidiană dintre forma x și forma medie m_k (sau centrul de greutate) al clasei k, de cardinal n_k:

$$d(x, m_k) = \|x - m_k\|, m_k = \frac{1}{n_k} \sum_{x \in F_k} x.$$

Fie o partiționare oarecare a mulțimii F în m clase și fie E valoarea criteriului aferent acestei partiționări. În continuare, să considerăm modificarea care se produce asupra acestei valori E în cazul în care se modifică clasificarea unei forme x de la clasa i la clasa j. Noua valoare E' aferentă noii partiționări va fi:

$$E' = E - \left\| x - \frac{n_i \cdot m_i - x}{n_i - 1} \right\|^2 + \left\| x - \frac{n_j \cdot m_j + x}{n_j + 1} \right\|^2 =$$
$$= E - \left(\left\| \frac{n_i}{n_i - 1} \cdot x - \frac{n_i}{n_i - 1} \cdot m_i \right\|^2 - \left\| \frac{n_j}{n_j + 1} \cdot x - \frac{n_j}{n_j + 1} \cdot m_j \right\|^2 \right)$$
$$= E - \left(\left(\frac{n_i}{n_i - 1} \right)^2 \cdot \|x - m_i\|^2 - \left(\frac{n_j}{n_j + 1} \right)^2 \cdot \|x - m_j\|^2 \right) =$$
$$= E - \Delta E(x)$$

Pentru a urmări minimizarea criteriului E, va trebui aleasă acea clasă j care să maximizeze $\Delta E(x)$ sau echivalent, să minimizeze cel de-al doilea termen al expresiei $\Delta E(x)$:

$$e_j = \left(\frac{n_j}{n_j + 1} \right)^2 \cdot \|x - m_j\|^2,$$

primul termen fiind constant pentru o aceeași valoare a lui x aparținând inițial clasei i.

Algoritmul bazat pe metoda celor mai mici pătrate este prezentat în continuare:

Se alege o partiționare inițială $F_0, F_1, \ldots, F_{m-1}$ *a mulțimii* F.
Se calculează valoarea E *și centrele de greutate* m_k *ale claselor,* $k=0, \ldots, m-1$.
repetă
  $clasificat \leftarrow 1$
  pentru $s \leftarrow 0, p-1$ **execută**
    fie F_i *clasa de care aparține forma* x_s
    fie $j \in \{0, 1, \ldots, m-1\}$ *pentru care* e_j *este minim*

dacă i ≠ j **atunci**
$F_i \leftarrow F_i - \{x_s\}$
$F_j \leftarrow F_j + \{x_s\}$
se actualizează centrele de greutate m_k
clasificat ← 0
sfârșit dacă
sfârșit pentru
până când clasificat = 1

Rezultatul furnizat de algoritm depinde de ordinea de parcurs a mulțimii formelor *F* precum și de partiționarea inițială, având tendința de a se bloca într-o situație de minim local. Rezultate mai bune se pot obține cu o metodă des utilizată și foarte simplu de implementat, prezentată în continuare.

Metoda nucleelor dinamice

Metoda nucleelor dinamice (cunoscută și sub denumirile de metoda celor *k* medii sau metoda *ISODATA*) constă în clasificarea unei forme *x* la acea clasă *k* pentru care distanța dintre forma *x* și centrul de greutate m_k al clasei este minimă. Procedeul este repetat până când structura claselor rămâne neschimbată.

Algoritmul este prezentat în continuare:

Se aleg m forme din mulțimea F *drept centrele de greutate* m_k *ale claselor,* k=0, ..., m-1.
repetă
clasificat ← 1
pentru s ← 0, p-1 **execută**
fie F_i *clasa de care aparține* x_s
fie j∈{0, 1, ..., m-1} *pentru care* $d(x_s, m_j)$ *este minimă*
dacă i ≠ j **atunci**
$F_i \leftarrow F_i - \{x_s\}$
$F_j \leftarrow F_j + \{x_s\}$
clasificat ← 0
sfârșit dacă
sfârșit pentru
se actualizează centrele de greutate m_k
până când clasificat = 1

Metoda prezintă la rândul ei anumite dezavantaje, rezultatele depinzând de:

- alegerea inițială a celor *m* centre de greutate. Pentru a înlătura această problemă, algoritmul poate fi reluat cu seturi diferite de puncte de plecare;
- distanța *d* folosită (a se vedea discuțiile de la distanțe privind normarea caracteristicilor cu abaterea medie pătratică);
- numărul *m* al claselor (pentru *m* egal cu numărul *p* al formelor se obține regula *NN*).

O variantă a acestei metode este reprezentată de actualizarea secvențială a centrelor de greutate ale claselor. Modificarea este preferabilă atunci când nu se dispune inițial de toate formele mulțimii *F*, acestea fiind obținute pe parcurs.

Algoritmul este prezentat în continuare:

Se aleg primele m *forme drept centrele de greutate* m_k *ale claselor,* k=0, ..., m-1.
$n_k \leftarrow 1$, k=0, ..., m-1
repetă
fie x *noua formă ce trebuie clasificată*
fie j∈{0, 1, ..., m-1} *pentru care* $d(x, m_j)$ *este minimă*
$F_j \leftarrow F_j + \{x\}$
$m_j \leftarrow (n_j * m_j + x) / (n_j + 1)$
$n_j \leftarrow n_j + 1$
până când <condiție_oprire>

Clasele produse de metodele prezentate mai sus alcătuiesc o așa-numită partiție "grea" a mulțimii *F* a formelor, în sensul că fiecare formă x_i din *F* aparține sau nu unei anumite clase *j*.

În contrast cu partițiile "grele" se află partițiile *fuzzy* caracterizate prin faptul că o formă x_i poate avea grade de apartenență diferite la clase diferite. În continuare, se va nota prin $u_{i,j}$ gradul de apartenență al unei forme x_i la clasa *j* considerat a fi de forma:

$$u_{i,j} = \frac{\exp\left(-\|x_i - m_j\|^2\right)}{\sum_{k=0}^{m-1} \exp\left(-\|x_i - m_k\|^2\right)}.$$

Se observă că pentru orice formă x_i este îndeplinită relația:

$$\sum_{j=0}^{m-1} u_{i,j} = 1.$$

Acest model este mai apropiat de modul de funcționare al clasificatorului uman, existând situații când și acesta ia decizii de tipul: o formă aparține unei clase într-o proporție de 70% și altei clase în proporție de 30%.

Cu considerațiile de mai sus, centrele de greutate ale claselor se vor calcula cu ajutorul mediei ponderate, astfel:

$$m_k = \frac{\sum_{x_i \in F_k} u_{i,k} \cdot x_i}{n_k}.$$

Algoritmul care implementează abordarea *fuzzy* a nucleelor dinamice este identic cu cele prezentate mai sus.

Metode ierarhice de clasificare

Metodele ierarhice de clasificare au avut drept punct de plecare taxonomia biologică, propunând o serie de grupări (divizări) între clase pe baza unor similarități (disimilarități) inter-clasă. Aceste metode pot fi clasificate în două categorii:

- metode descendente (divizive) ce au drept punct de plecare o singură clasă formată din toate formele mulțimii *F* care este divizată recursiv;
- metode ascendente (de aglomerare) care procedează la gruparea succesivă a claselor, având inițial un număr de clase egal cu cardinalul mulțimii *F*.

Un algoritm aparținând celei de-a doua categorii de metode este prezentat în continuare. Numărul de clase este reprezentat de variabila *m*. Prin d_{ij} s-a notat distanța inter-clasă dintre clasele *i* și *j*.

Se formează clasele `Fk={xk}, k=0, ..., p-1`
Se calculează distanțele d_{ij}
$m \leftarrow p$
repetă
  fie $i, j \in \{0, 1, ..., m-1\}$ *pentru care* d_{ij} *este minimă*
  $F_i \leftarrow F_i \cup F_j$
  $m \leftarrow m-1$
  Se recalculează distanțele d_{ij}
până când <condiție_oprire>

Condiția de oprire poate fi determinată de: atingerea unei anumite valori *m* pentru numărul de clase, situarea tuturor distanțelor d_{ij} sub o anumită valoare prag etc.

Distanțele inter-clasă d_{ij} pot fi calculate folosind una dintre următoarele formule:

$$d_{i,j} = \max\{d(x,y) / x \in F_i, y \in F_j\}$$
$$d_{i,j} = \min\{d(x,y) / x \in F_i, y \in F_j\}$$
$$d_{i,j} = d(m_i, m_j)$$
$$d_{i,j} = \frac{n_i \cdot n_j}{n_i + n_j} \cdot d(m_i, m_j)$$

Semnificația variabilelor care apar în aceste formule este următoarea:

- *d* reprezintă o distanță oarecare (de exemplu, distanța euclidiană);
- m_i și m_j reprezintă centrele de greutate ale celor două clase *i* și *j*;
- n_i și n_j reprezintă numărul de elemente ale celor două clase *i* și *j*.

Se poate observa similitudinea algoritmului prezentat anterior cu algoritmul de determinare a arborelui de cost minim al unui graf având drept noduri cele *p* clase inițiale și drept muchii distanțele inter-clasă. La fiecare pas, se adaugă un nou nod în arborele de cost minim.

Plecând de la această similitudine, să considerăm în continuare un graf neorientat (numit ***graful formelor***) având un număr *p* de noduri corespunzătoare celor *p* forme, iar matricea de adiacență $A_{p\times p} = (a_{ij})_{p\times p}$ definită astfel:

$$a_{i,j} = \begin{cases} 1, \text{pentru } d(x_i, x_j) \le D \\ 0, \text{altfel} \end{cases},$$

unde *d* este o distanță oarecare, iar *D* este o valoare de prag.

Cu aceste considerente, rezultă un nou algoritm de clasificare a formelor mulțimii *F*, bazat pe determinarea componentelor conexe ale grafului construit cu specificațiile de mai sus.

Procesul de împărțire al formelor în clase este controlat prin valoarea de prag *D*.

Analiza comparativă a clasificatorilor

Evident, modelul de clasificator va fi ales în funcție de tipul aplicației, numărul și tipul caracteristicilor, mărimea setului de învățare, existența unor informații prealabile privind distribuția formelor etc.

În continuare se vor prezenta performanțele obținute cu o serie de clasificatori pentru o aplicație de recunoaștere a formelor. Trebuie precizat faptul că discuțiile ce vor urma au drept scop doar punerea în evidență a metodelor practice de construire a unor clasificatori și nu au pretenția de a rezolva problema specifică avută în vedere.

Având considerațiile făcute, fie următoarea aplicație de recunoaștere a formelor având drept obiectiv: recunoașterea *corpusculilor Danne* din imagini de microscopie electronică a serului bolnavilor de *hepatită virală B*. O astfel de imagine este prezentată în figura 1.

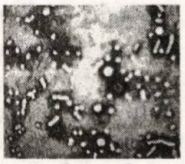


Figura 1

Imaginea a fost preluată din [7] (pag. 298). *Virusul hepatic B* este considerat a fi reprezentat de particula virală numită *corpuscul Danne*. După cum se observă și din imagine, în serul bolnavilor de *hepatită virală B* s-au pus în evidență în microscopie electronică trei forme ale *antigenelor virale B*: particule sferice mici cu un diametru de 22 nm, forme tubulare de grosime 22 nm și lungime 20-250 nm și *corpusculi Danne* de formă circulară cu un diametru de 42 nm.

Figura 2

Prin utilizarea unor tehnici specifice procesării imaginilor (filtrare, segmentare etc.) se obține o imagine prelucrată, binară, prezentată în figura 2, care permite identificarea celor trei tipuri de particule cu ajutorul unui algoritm de determinare a componentelor conexe.

Pentru caracterizarea celor trei tipuri de forme se vor folosi două caracteristici: circularitatea și aria.

Circularitatea este definită ca fiind proporțională cu raporul dintre aria și pătratul perimetrului unei figuri geometrice: $C = \frac{4 \cdot \pi \cdot A}{P^2}$, unde *A* reprezintă aria figurii, iar *P* perimetrul. *Circularitatea* are valoarea 1 pentru un disc și o valoare subunitară pentru orice altă figură geometrică. Ea va distinge dintre cele două forme circulare și formele tubulare, iar aria va realiza discriminarea dintre cele două forme circulare (de diametre 22 și respectiv 42 nm).

În figura 3 sunt reprezentate formele în spațiul celor două caracteristici (circularitatea și aria).

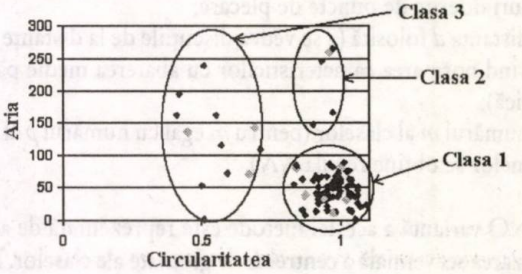


Figura 3

Din această reprezentare se observă gruparea relativă a formelor din cele trei ($m = 3$) clase și anume: clasa 1 - corpusculii circulari mici 22 nm (arie mică și circularitatea ≈ 1), clasa 2 - *corpusculii Danne* 42 nm (arie mare și circularitate ≈ 1), clasa 3 - corpusculii de formă tubulară (aria variază datorită lungimii lor între 20-250 nm, iar circularitatea ≈ 0.5).

Pentru contextul învățării supravegheate, cele p (p = 110) forme au fost împărțite în două seturi: setul de învățare (100 de puncte - punctele negre din planul caracteristicilor) și setul de test (10 puncte - punctele gri). În tabelul 1 sunt prezentate avantajele și dezavantajele folosirii câtorva clasificatori aparținând contextului supravegheat al recunoașterii formelor.

În figura 4 sunt prezentate rezultatele utilizării clasificatorilor bazați pe *NN*, *KNN* ($k = 3$) și funcții discriminant pentru câteva elemente aparținând mulțimii de test.

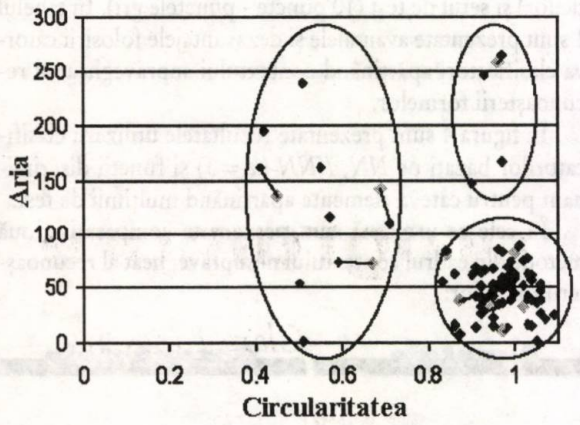
În cele ce urmează sunt prezentate comparativ două metode din cadrul contextului nesupravegheat al recunoașterii formelor:

Clasificator	Complexitate		Avantaje	Dezavantaje
	învățare	clasificare		
Nearest Neighbour(NN)	n/a^*	$O(n \times p)$	• Implementare ușoară, rapidă. • Nu necesită învățare și nici alte analize efectuate asupra setului de învățare.	• Este necesară memorarea setului de învățare. • Timp mare de calcul (este parcurs întreg setul de învățare pentru clasificarea unei singure forme). • Sensibilitatea la zgomot. • Sensibilitatea la distanța folosită.
K Nearest Neighbours (KNN)	n/a^*	$O(k \times n \times p)$	• Implementare rapidă. • Performanțe mai bune decât *NN* fiind mai puțin sensibil la zgomot.	• Este necesară memorarea întregului set de învățare. • Timp de clasificare mult prea mare.
Funcții discriminant liniare	$O(n \times p)$	$O(m \times n)$	• Nu este necesară memorarea setului de învățare. • Timp scăzut de clasificare comparativ cu celelalte două metode.	• Pot fi aplicate doar în cazul în care clasele sunt liniar separabile. • Necesită memorarea a $n \times (m + 1)$ coeficienți.
Clasificatori Bayes	$O(p)$	$O(m \times n)$	• Minimizează rata medie a erorii.	• Sunt necesare informații privind $Pr\{x\|j\}$ și $Pr\{j\}$.
Perceptronul multistrat (3 straturi)	variabilă**	$O(h \times (n + m))$	• Performanțe superioare comparativ cu celelalte metode. • Posibilități foarte bune de generalizare. • Suprafețe de separare complexe.	• Determinarea numărului h de neuroni din stratul ascuns se face experimental. • Performanțele depind fundamental de setul de învățare. • Există posibilitatea ca rețeaua să se oprească într-un optim local. • Există posibilitatea ca rețeaua să învețe foarte bine ("pe de rost") formele din mulțimea de învățare ceea ce conduce la o slabă generalizare.

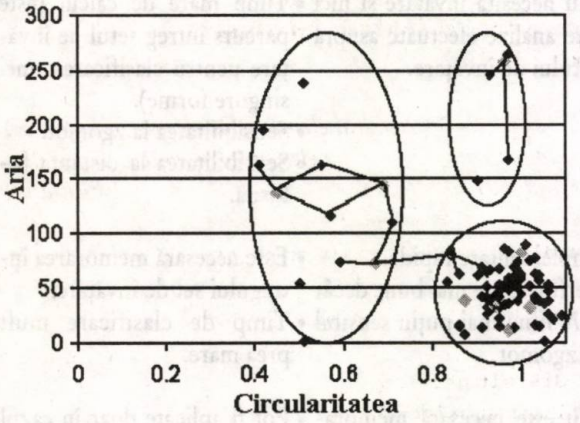
* *Nu se aplică în acest caz.*

** *Depinde de: mărimea p a setului de învățare, de rata de eroare ce se dorește a fi obținută, etc.*

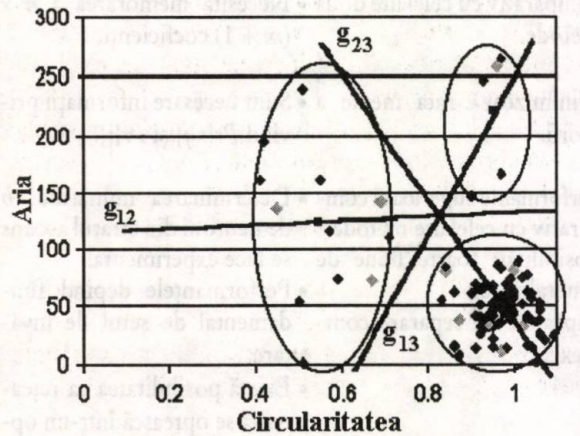
Tabelul 1: Analiza comparativă a clasificatorilor în învățarea supravegheată



a) regula *NN*



b) regula *KNN (k = 3)*



c) funcții discriminant. Clase separate de suprafețele de separare g_{12}, g_{13}, și g_{23}.

Figura 4

- *metoda nucleelor dinamice*

Complexitatea algoritmului de clasificare depinde de concentrarea formelor din setul de învățare și numărul *m* de clase. Are ca avantaj minimizarea inerției *intra-clasă*, iar ca dezavantaj faptul că rezultatele depind de valorile inițiale, distanța utilizată și numărul *m* al claselor.

- *metoda grafului formelor*

Complexitatea algoritmului de clasificare este $O(p^2 \cdot n)$. Printre avantajele acestei metode se află faptul că permite alegerea valorii de prag *D* și clasele obținute sunt omogene.

Timpul de calcul mare pentru un număr mare *p* de forme aparținând setului de învățare constituie unul dintre principalele dezavantaje ale utilizării metodei grafului formelor.

În figura 5 este ilustrată aplicarea metodei nucleelor dinamice prin conturarea celor trei clase. Formele sunt grupate în jurul centrelor de greutate.

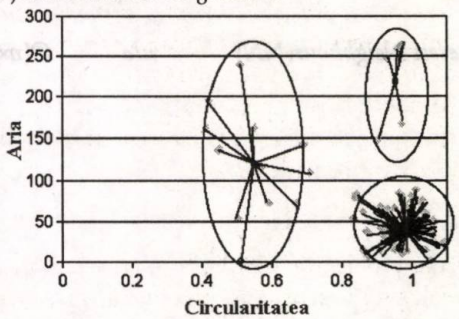


Figura 5

Concluzii

Acest serial s-a dorit a fi o introducere în recunoașterea statistică a formelor, prezentând câteva dintre tehnicile și metodele utilizate în cadrul celor două tipuri de abordări: învățarea supravegheată și nesupravegheată.

Cu toate că exemplele abordate au avut în vedere aspecte specifice procesării imaginilor, recunoașterea formelor include un domeniu mult mai vast al aplicațiilor permițând, după cum s-a prezentat, clasificarea și distingerea dintre obiecte și fenomene supuse unui același model matematic abstractizat.

Bibliografie

1. Duda R., Hart P., *Pattern Classification and Scene Analysis*, John Wiley & Sons, 1973
2. Belaid A., Belaid Y., *Reconnaissance des formes: Methodes et applications*, InterEditions, 1992
3. Milgram M., *Reconnaissance des formes: Methodes numeriques et connexionnistes*, Armand Colin, 1993
4. Pentiuc Șt. Gh., *Aplicații ale recunoașterii formelor în diagnosticul automat*, Ed. Tehnică, 1997
5. Sonka M., Hlavac V., Boyle R., *Image Processing, Analysis and Machine Vision, 2nd Edition*, PWS Publishing, 1998
6. Dumitrescu D., Hariton Costin, *Rețele neuronale: Teorie și aplicații*, Teora, 1996
7. Georgescu L., Tudose N., Potencz E., *Morfopatologie*, Editura Didactică și Pedagogică, București, 1982

Radu-Daniel Vatavu este student în anul IV la Universitatea Ștefan cel Mare din Suceava. Poate fi contactat prin e-mail la **raduvro@yahoo.com**.

TEOREME de programare

Clara Ionescu

În numerele precedente ale revistei am prezentat primele două "teoreme" care ofereau modele pentru prelucrarea unui șir de date, respectiv pentru stabilirea unei proprietăți a unui element dintr-un șir sau a unei proprietăți globale a șirului. Aceste modele sunt utile deoarece respectarea lor garantează corectitudinea subalgoritmului respectiv și conferă un plus de claritate codului.

Selecția

Rezolvările problemelor P5-P8, prezentate în contextul teoremei de programare pe care am numit-o *Decizie*, verifică o anumită proprietate într-un șir de date și comunică prezența sau absența ei. Analizând mai atent rezolvările, constatăm că am putea fructifica subalgoritmul, afișând în plus informații cum ar fi poziția pe care se află elementul cu proprietatea cerută. Fie în continuare problemele:

P9. Cunoscând denumirea unei luni, să se afișeze numărul ei de ordine!

P10. Să se determine cel mai mic divizor diferit de 1 al unui număr natural, strict mai mare decât 1!

P11. Cunoscând onomasticele (zilele prenumelor) dintr-un calendar, să se afișeze data la care își serbează ziua o persoană având un prenume dat!

Evident, din nou ne punem întrebarea, ce anume au în comun enunțurile de mai sus? Datele de intrare formează un șir, iar ceea ce se mai dă, este - cu siguranță - un element din șirul datelor.

Rezultă, în primul rând, că de data aceasta nu trebuie să verificăm apartenența acestei date la șirul dat, pentru că enunțul garantează acest lucru. În P9 se dă denumirea unei *luni*, iar în P11, un *prenume*. Dar ce putem spune despre această "garanție" în cazul problemei P10? Aici șirul datelor care trebuie prelucrate (divizorii) nu trebuie citit, deoarece poate fi generat. În plus, știm că orice număr natural mai mare decât 1 (deci și numerele prime) are cel puțin doi divizori: 1 și numărul însuși. Deci, dacă numărul *n* nu se divide cu nici un număr mai mic decât *n*, dar este mai mare sau egal cu 2, totuși are un divizor diferit de 1, și anume chiar pe *n*.

Vom prezenta în continuare câteva detalii privind declarațiile din programul care va conține subalgoritmul care rezolvă o asemenea problemă:

```
n: întreg                {numărul elementelor șirului de prelucrat}
x: tablou(1..n):         tip element{elementele șirului dat}
nr_ord: întreg                                        {rezultatul}
```

```
Subalgoritm Selecția(n,x,nr_ord):
  nr_ord := 1
  Cât timp nu P(x[nr_ord]) execută
    nr_ord := nr_ord + 1
  Sfârșit cât timp
Sfârșit subalgoritm
```

Nu am verificat dacă valoarea contorului a depășit dimensiunea șirului dat, tocmai datorită faptului că enunțul garantează existența în șir a valorii căutate. Elementul cu proprietatea cerută va fi găsit cel târziu atunci când prelucrăm ultimul element.

Vom aplica această rezolvare generală pentru problemele enunțate la începutul articolului, dând implementările în limbajul *Pascal*.

Problema P9 cere numărul de ordine al unei luni date.

```
procedure P9(n:Byte; luni:Tluni; cuv: string;
                                        var nr:Byte);
{avem definit tipul Tluni = array[1..12] of string}
begin
  nr:=1;
  while luni[nr]<>cuv do                { avansăm în șir }
    Inc(nr);                  { din while am ieșit în momentul }
end;                                  { în care am găsit luna dată }
```

P10. Această problemă cere determinarea celui mai mic divizor, diferit de 1, al unui număr natural. Vom prezenta o posibilă implementare în care se folosește o funcție, și nu o procedură.

Evident, și în cazul problemei precedente am fi putut lucra cu o funcție.

```
function divizor(n: Word): Word;
var i:Word;
begin
  i:=2;                          {primul divizor "permis"}
  while n mod i <> 0 do
    Inc(i);                      {următorul divizor posibil}
  divizor:=i
end;
```

Acum, ne putem întreba dacă este sau nu cazul să "generăm" valori pentru divizorii "posibili" care sunt mai mari decât rădăcina pătrată a numărului dat? Evident, ***nu***! Rezultă că în condiția din **while**, de data aceasta este indicat să adăugăm și subexpresia `(i ≤ rad)`, unde valoarea `rad` trebuie calculată înaintea ciclului, pentru a nu cere evaluarea ei de fiecare dată când acesta se reia. Va trebui să fim atenți pentru cazul în care numărul *n* este prim. Dacă nu a fost găsit nici un divizor mai mic decât rădăcina pătrată a lui *n*, atunci *n* este număr prim, deci rezultatul va fi *n*.

P11. Vom determina data din an (luna și ziua) la care o persoană, având prenumele cunoscut, își va serba ziua numelui. Calendarul constituie șirul *x* și presupunem că acesta cuprinde *n* articole. Articolele vor avea structura prezentată mai jos:

```
type onomastica=record
                  prenume:string;
                  luna,ziua:Byte
                    end;
     calendar=array[1..MaxN] of onomastica;
var x:calendar; n:Byte;
    pren:string; zi:onomastica;

procedure P11(n:Byte;x:calendar;pren:string;
                                var zi:onomastica);
var i:Byte;
begin
  i:=1;
  while x[i].prenume<>pren do Inc(i);
  zi:=x[i]
end;
```

În această implementare nu am putut respecta fidel modelul dat pentru selecție, deoarece a apărut o diferență: nu numărul de ordine al unui element din șirul dat ne-a interesat, ci câmpurile acestuia. Cu toate că ar fi trebuit să păstrăm doar valorile cîmpurilor care diferă de cel a cărui valoare o cunoșteam în prealabil, în procedură, am păstrat în parametrul de ieșire `zi` întregul element de tip **record**, pentru a nu lungi lista parametrilor.

Se observă că acest model de rezolvare conduce la găsirea primului element, având proprietatea cerută. Dacă ne interesează ultimul astfel de element vom parcurge șirul în sens invers, adică începând cu ultimul și "avansând" spre primul.

Căutarea (secvențială)

Problema **P6** a fost rezolvată folosind modelul oferit de *Decizie*. Aceasta era, de fapt, o simplă căutare (un cuvânt dat este sau nu denumirea unei luni din an?). Acolo era suficient să răspundem la întrebări de genul "*există*?". Revenim, tratând tipul de problemă în care cele două cerințe din ultimele două clase de probleme se combină, respectiv, după aflarea răspunsului la întrebarea "*există*?", afișăm, de exemplu, poziția pe care se află elementul căutat. Cu alte cuvinte, în această clasă intră probleme care nu garantează existența elementului căutat în șirul dat.

P12. Se consideră valorile intrărilor și ieșirilor sumelor de bani care reprezintă încasările și plățile efectuate de către o firmă pe parcursul unei luni. În cazul în care există o zi cu pierderi, să se afișeze această zi! (Deci, se cere ziua în care plățile au fost mai mari decît încasările.)

Evident, modelul de rezolvare se bazează pe modelele prezentate până acum.

```
Subalgoritm Căutare(n,x,găsit,nr_ord):
  i:=1
  Cât timp (i<=n) și nu (P(x[i]) execută i:=i+1
  Sfârșit cât timp
  găsit:=i<=n
  Dacă găsit atunci nr_ord:=i
Sfârșit subalgoritm
```

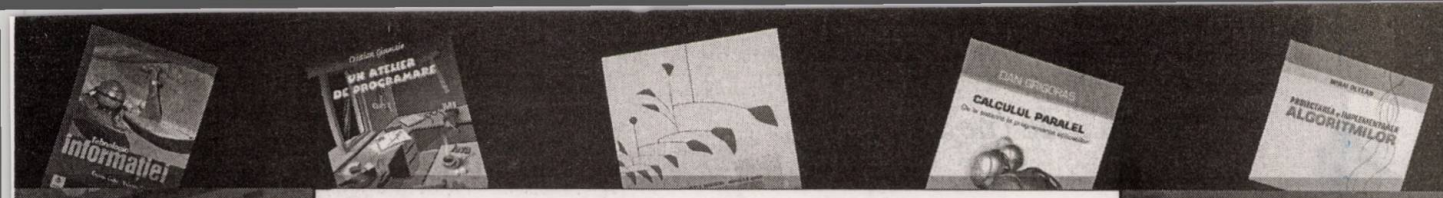
Se observă că soluția pornește cu modelul *Deciziei* care se completează cu determinarea rezultatului conform *Selecției*.

În rezolvarea problemei **P12**, presupunem că `int` și `ies` sunt două tablouri având numărul de elemente egal cu numărul zilelor din luna prelucrată, deci în variabila `ziua` vom păstra indicele primei zile din lună în care firma a avut pierderi, dacă există o asemenea zi. Evident, înainte de afișare, în unitatea de program apelantă vom testa valoarea variabilei `găsit`, transmisă ca parametru de ieșire (prin referință):

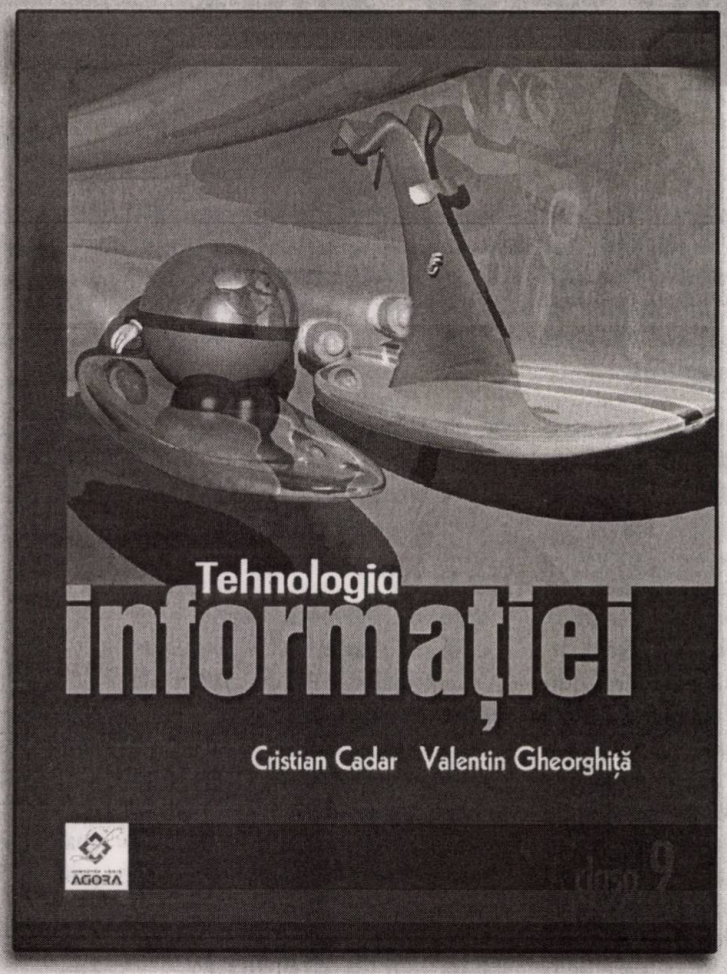
```
procedure P12(n:Byte; int,ies:tablou;
              var gasit:Boolean; var ziua:Byte);
var i:Byte;
begin
  i:=1;
  while (i<=n) and (int[i]>=ies[i]) do
    Inc(i);
  gasit:=i<=n;
  if gasit then ziua:=i
end;
```

Din nou, dacă ne interesează ultima zi cu pierderi, vom parcurge tablourile în sens invers.

În următorul articol vom vedea cum procedăm dacă dorim să determinăm toate elementele unui șir dat, având o proprietate dată.



COMPUTER LIBRIS AGORA



Lucrarea de față își propune să pregătească cititorul în scopul utilizării unui calculator personal. Ea urmărește programa școlară, aprobată de Ministerul Educației Naționale, pentru disciplina "Tehnologia informației", disciplină în planul de învățământ a claselor a IX-a la profilul teoretic și pedagogic. Manualul corespunde și programei de la disciplina "Informatică - Tehnologii asistate pe calculator", profilul tehnologic.

Pentru parcurgerea acestei cărți nu sunt necesare cunoștințe prealabile în domeniu, termenii și noțiunile prezentate sunt explicate detaliat în carte. Exemplele sunt sugestive, cartea nu este încărcată inutil cu informații care depășesc programa amintită.

COMPUTER LIBRIS AGORA
Informații la tel. 064-192422

COMPUTER LIBRIS AGORA

După ONI 2002

Mihai Scorțaru

Părerile despre Olimpiada Națională de Informatică din acest an au fost, ca de obicei, împărțite. Cei care au vrut să privească jumătatea plină a paharului susțin că totul a fost OK, cei care privesc jumătatea goală cred că a fost un dezastru. Sunt puțini cei care pot privi ambele jumătăți... Nici eu nu pot! În acest articol voi prezenta o părere personală referitoare la latura științifică a acestei olimpiade.

Totul a început în vacanța de vară din acest an. Era și *GInfo* în vacanță... dar mă gândeam, din când în când, ce ar trebui să conțină primul număr din acest an școlar. Nu a fost prea greu să îmi dau seama că vreți să apară cât mai multe probleme propuse și, în special, cât mai multe rezolvări. Cei drept, m-au ajutat și formularele pe care le-ați completat pentru a participa la ediția 2001/2002 a concursului nostru. Și cum în numărul din mai fuseseră publicate enunțurile de la *ONI* 2002, era destul de evident că în secțiunea dedicată problemelor rezolvate ar trebui să apară soluțiile problemelor de la olimpiadă.

După mai multe săptămâni am luat în sfârșit *CD*-ul pe care scria *ONI* 2002 și am început să studiez rezolvările. Vă mărturisesc că nu am înțeles mare lucru! Nici nu prea aveam cum. *CD*-ul era structurat foarte frumos, pentru fiecare problemă apărea enunțul, datele de test, datele de ieșire corecte, un evaluator și o prezentare a soluției oficiale. Prima problemă era că, din când în când, unele mai lipseau; totuși nu a fost cea mai mare dificultate cu care m-am confruntat deoarece unele prezentări ale soluției oficiale nu ofereau practic nici un indiciu referitor la modul de rezolvare. Cel mai bun exemplu în acest sens este problema de la clasele a XI-a și a XII-a în care se cerea "ghicirea" unei permutări. Prezentarea soluției oficiale este următoarea:

1. *Se determină poziția corectă a unui element din permutare în $O(N^2)$ întrebări.*
2. *Utilizând elementul fixat, putem determina elementul care apare pe orice altă poziție în $O(N)$ întrebări.*

Complexitate: $O(N^2)$ apeluri ale funcției CHECK $\Rightarrow O(N^3)$.

Cât se poate de clar, nu? Ca să nu mai spun că, până la urmă s-a dovedit că poziția corectă a unui element se poate determina în $O(N)$ "întrebări". Oricum, voi reveni asupra corectitudinii soluțiilor oficiale...

Rezolvările din GInfo

A devenit clar destul de repede faptul că astfel de soluții nu pot fi prezentate în revista noastră. Se pare că m-am înșelat pentru că tocmai ați citit una dintre ele... Dacă vreți și alte monstre, încercați să faceți rost de *CD*-ul magic pe care scrie *ONI* 2002.

În plus, mi-am adus aminte că există persoane care nu vor să își vadă "operele" publicate în *GInfo* și am putea avea probleme legate de drepturile de autor. Trebuie să știți că, în principiu, enunțurile sunt publice, deci pot fi publicate de oricine, dar cu rezolvările treaba este mai complicată. În fine, sper să nu am probleme datorită publicării "rezolvării" de mai sus!

Așadar, trebuia să "inventez" rezolvări care să difere în proporție de cel puțin 60% (potrivit legii, dar cifra ar putea să nu fie exactă) față de cele pe care le-am găsit pe *CD*. Și-ntr-o altă zi de vară am început... cu clasa a IX-a că, am zis eu, o fi mai simplu.

Astfel au apărut rezolvările "geniale" din ultimele două numere ale revistei noastre. Dacă ați găsit greșeli, atunci vina pentru acestea îmi aparține în totalitate. Membrii comisiilor de la *ONI* 2002 nu le-au văzut înainte de apariția în *Ginfo* și nu și-au putut exprima nici o opinie referitoare la ele.

Clasa a IX-a

După cum am spus, am început cu clasa a IX-a, mai exact cu problema ***Pentagon***, cea care i-a cam enervat pe unii americani care au aflat de ea. Și citesc eu o matrice, o parcurg pe coloane, număr blocurile de diferite lungimi și scriu rezultatul. Într-un sfert de oră termin și, mulțumit că într-adevăr a fost o problemă simplă, încep să configurez ***GInfo Evaluator*** **0.5** ***Beta*** ca să o testez. La mai puțin de 10 secunde după apăsarea butonului ***Evaluate*** îmi studiez punctajul și constat mirat că nu e de 100 de puncte. Am pierdut cam o oră încercând să-mi găsesc greșeala, dar nu am reușit. Studiam cele aproape 50 de linii de cod și nu vedeam nici o greșeală. Era totuși des-

tul de greu să o găsesc, mai ales că nici nu exista. Am început să găsesc greșeli imaginare și să le "corectez", dar punctajul nu voia deloc să se apropie de 100 de puncte; mai mult, de obicei era 0. Evaluam cu venerație soluția oficială și mă uitam la numărul 100 care apărea de fiecare dată în dreptul punctajului. Într-un moment de inspirație am comparat un fișier greșit furnizat de programul meu cu cel furnizat de programul comisiei. Am observat că unele numere din fișierul meu erau mai mari decât numerele corespunzătoare din celălalt fișier. Într-un alt moment de inspirație am efectuat niște scăderi; diferențele, acolo unde existau, erau 256 sau 512. În fine, în al treilea moment de inspirație m-am uitat în sursa comisiei și am văzut un "surprinzător" `array of Byte`. Nu mai aveam nevoie de inspirație. Am modificat tot ce era `Byte` în `Integer` și am rulat din nou programul. Ușurat, am observat că acum rezultatele furnizate de cele două programe erau identice. Așadar și prin urmare programul comisiei era perfect, dar conținea o eroare minoră. Efectele nu au fost prea grave: ce contează dacă doar câțiva concurenți care au rezolvat corect problema nu au primit punctele meritate? Trebuia să se gândească și ei că cei din comisie pot greși și nu aveau decât să ghicească greșeala și să scrie un program cu exact aceeași greșeală. Doar se spune că trebuie să ai și noroc pentru a câștiga la olimpiadă...

Am trecut la următoarea problemă: ***Pod***. Am început să studiez soluția oficială. Era o programare dinamică simplă, dar la clasa a IX-a... E adevărat că în prezentarea soluției nu apărea deloc termenul *programare dinamică*. De data aceasta am trecut direct la configurarea evaluatorului, am compilat cele trei fișiere sursă ale comisiei și am început să testez. Nu am reușit să-mi dau seama exact care dintre ele reprezintă soluția "adevărată". Probabil că nici unul fiindcă nu au reușit nicicum să obțină 100 de puncte. Am trecut peste asta și am implementat soluția proprie. Am luat 100 de puncte "din prima" și am trecut fericit mai departe. Nu pot spune decât că cei din comisie trebuie doar să inventeze probleme, nu neapărat să le și rezolve. Am totuși rezerve față de această problemă. S-ar putea să nu fi testat eu așa cum trebuie sau *CD*-ul pe care l-am avut să fi fost inscripționat greșit!

La cea de-a treia problemă (***Suma***) nu am întâmpinat nici o dificultate, dar nu am mai fost la fel de norocos la cea de-a patra (***Becuri***). În primul rând soluția oficială s-ar putea să fi fost extraordinar redactată, dar lipsea cu desăvârșire. În plus, am constatat că programul folosit pentru verificarea soluțiilor concurenților conținea o mică eroare. În enunț se preciza că dacă nu trebuie comutată nici o linie sau nici o coloană, linia corespunzătoare din fișierul de ieșire trebuie să conțină un 0. Evaluatorul accepta ca fiind corecte și fișiere de ieșire în care linia respectivă era vidă. De această dată eroarea a fost în favoarea concurenților, ei putând primi puncte nemeritate pentru anumite teste. Din nou treaba cu norocul la olimpiadă... Mai face comisia și cadouri...

La celelalte două probleme, ***Discuri*** și ***Cod***, nu am găsit nici o greșeală și sper că nici nu au fost!

Clasa a X-a

Într-o altă zi am trecut la clasa a X-a. Era prea mult să rezolv peste șase probleme într-o zi așa că am luat o pauză care, evident, a durat o perioadă. Cu gândul la ce am pățit la clasa a IX-a am început să studiez problema ***Hotel***. Nu am priceput mai nimic din ce se spunea în prezentarea soluției. În plus, mi se părea mult prea complicat așa că am renunțat. Am observat că, de fapt, problema cerea determinarea numărului aranjamentelor de n luate câte k. Am scris un program care folosea această formulă și am primit 100 de puncte. Sursa mea era, evident, mult mai scurtă decât cea a comisiei, iar explicațiile mult mai clare (în opinia mea). De data aceasta comisia "și-a făcut capul mare singură" sau a considerat că elevii de clasa a X-a nu cunosc formula de calcul a aranjamentelor.

A urmat problema ***Lac*** pe care am rezolvat-o repede, dar evaluatorul îmi acorda doar 90 de puncte. Am trecut cam prin aceleași faze ca la problema ***Pentagon***, dar de data aceasta nu am găsit greșeli la implementarea soluției oficiale. Într-un târziu am găsit greșeala; era în evaluator! Programul de verificare a corectitudinii soluției concurenților se oprea, la testul raportat ca fiind eronat, cu eroarea `Stack overflow`. Am modificat puțin programul de verificare astfel încât să folosească mai puțină memorie în timpul testării și eroarea nu a mai apărut. Deci, o nouă eroare a comisiei... Dacă un concurent avea ghinionul ca fișierul său de ieșire, deși corect, să determine apariția acestei erori la rularea programului de verificare, atunci punctele erau pierdute. Din nou șansa își spune cuvântul... Concurenții nu aveau decât să ghicească cum va fi realizat evaluatorul și să aibă grijă ca fișierele lor de ieșire să nu ducă la întreruperea execuției acestuia.

Am trecut la problema ***Foto***, una dintre cele mai interesante de la *ONI* 2002 (din nou, după părerea mea). Am văzut că se rezolva folosind metoda *backtracking* (cel puțin așa se susține în prezentarea soluției oficiale). Cum mie nu îmi place deloc această metodă și o folosesc doar în cazuri extreme (așa cum ar trebui să facă toată lumea - tot după părerea mea), am încercat să găsesc o altă soluție. Nu a fost foarte greu și am reușit să obțin un algoritm liniar. Mi-am dat seama de-abia la sfârșit că programul meu era mai lung decât cel al comisiei; dar era la fel de corect și era departe de a utiliza metoda *backtracking*.

A urmat "confruntarea" cu ***Balanța***... Nu am de reproșat decât un aspect minor legat de eficiență. Soluția oficială parcurgea un șir de la -8000 la 8000. Există posibilitatea unor îmbunătățiri, pe care le-am prezentat și în *GInfo* 12/6.

În primul rând este suficient să se parcurgă șirul de la -6000 la 6000 ($15 \cdot 20 \cdot 20 = 6000$). Mai pot fi aduse alte îmbunătățiri dacă se iau în considerare distanța celui mai îndepărtat cârlig, numărul greutăților și suma greutăților. Sunt doar detalii, soluția oficială primind punctajul maxim.

Ultima problemă de la clasa a X-a a fost ***Aliniere***, problemă a cărei rezolvare corectă depindea, în opinia mea, de un aspect ridicol: doi soldați aflați în mijlocul șirului pot avea înălțimi egale dacă privesc în direcții diferite. Este o "capcană" în care cred că ar cădea aproape toți cei care încearcă să rezolve problema. Pe de altă parte, astfel de capcane fac parte din regulile jocului... Totuși, în soluția oficială algoritmul de determinare a eventualei perechi de soldați cu înălțimi egale are ordinul de complexitate $O(n^2)$; în soluția prezentată în numărul din octombrie al revistei noastre apare un algoritm liniar care realizează aceeași operație.

Clasele a XI-a și a XII-a

Până aici a fost cum a fost, dar am ajuns și la problemele cu care s-au confruntat elevii din clasele a XI-a și a XII-a. Vă mărturisesc că am vrut să le rezolv cât mai repede ca să găsesc o explicație a motivului pentru care punctajele obținute de concurenți au fost foarte mici. Și am început... cu problema ***Arbore***. Mai întâi m-am uitat peste sursele comisiei (erau două) și am văzut că una are peste 200 de linii, iar cealaltă peste 300. Mi-am zis că trebuie să fie și o problemă grea la olimpiadă (numărul de linii al codului nu indică neapărat acest lucru, dar mai sunt și alte aspecte pe care le voi descrie în continuare) și am început să o rezolv. Pe măsură ce trecea timpul, îmi dădeam seama tot mai mult că îmi cam pierdusem antrenamentul. Deși aveam în față prezentarea soluției, abia după șase ore de muncă am reușit să primesc 100 de puncte. Mai mult, apoi am mai găsit câteva greșeli "minore", astfel încât programul meu nu funcționa corect în anumite situații. Nici una dintre acele situații nu a fost "surprinsă" în datele de test folosite pentru evaluare. După încă două ore am reușit să ajung la ceea ce se numește soluția prezentată de *GInfo*. Apoi am stat și m-am gândit puțin și mi-am dat seama că, totuși, am mai multă experiență decât participanții de la *ONI* 2002, deci sunt mari șanse ca rezolvarea acestei probleme să nu fi fost posibilă pentru participanți, mai ales că aveau de rezolvat trei probleme în ziua de concurs. Nu pot să susțin că nu există participanți care ar putea rezolva problema mai repede decât mine; nu susțin nici că faptul că eu am rezolvat-o în șase ore citind prezentarea soluției este semnificativ, dar pot să spun că această problemă a fost mult prea grea pentru un concurs de acest nivel. Sunt gata să accept faptul că "mi-am ieșit din mână" sau nu mai sunt capabil să rezolv problemele de la olimpiadă dacă mi se arată că mă înșel. De fapt, abia aștept! N-am terminat cu această problemă. Iată cum se prezintă în soluția oficială modul în care este rezolvat un anumit nod x:

- *x are mai mult de 2 fii terminali → nu există soluție;*
- *x are exact 2 fii și aceștia sunt terminali → unesc cei doi fii printr-o muchie; am obținut un ciclu cu 3 vârfuri care îl "rezolvă" pe x și cei doi fii ai săi;*
- *x are 2 fii terminali dar și alte "fire": unesc cei doi fii terminali (rezultă un ciclu de lungime 3 în care intră x și cei doi terminali), apoi rezolv firele (unesc printr-o muchie primul și ultimul vârf de pe "fir"; evident acest lucru este posibil dacă toate firele au lungimea mai mare decât 2);*
- *x are 1 fiu terminal și evident alte "fire": unesc vârful terminal cu extremitatea finală a unuia dintre fire (cel de lungime 2 dacă există, oricare altul dacă nu există), apoi rezolv firele (acestea trebuie să aibă lungime mai mare decât 2);*
- *x nu are fii terminali, numai fire; rezolv firele (dacă există mai mult de 2 fire de lungime 2, nu există soluție), unind între ele extremitățile finale a două fire (cele de lungime 2 dacă există, sau oricare altele), iar restul le rezolv independent (unind extremitatea lor inițială cu extremitatea lor finală).*

Ce părere aveți? Nu s-ar putea spune, pur și simplu, că problema nu are soluție dacă există mai mult de două fire cu lungimea mai mare decât 2? Mai mult, nu se poate spune că se unesc extremitățile a celor mai scurte două fire și apoi celelalte fire sunt rezolvate individual? Părerea mea este că se poate! Și ajungem să avem un singur caz în loc de cinci! Pare ceva mai simplu, nu credeți? S-ar putea să mă înșel din nou, dar vă asigur că am obținut până la urmă 100 de puncte tratând un singur caz în loc de cinci care tratează, de fapt, același lucru.

Așadar, problema este foarte drăguță cu un enunț simplu și elegant dar, în opinia mea, este mult prea dificilă, iar prezentarea soluției lasă de dorit.

Urmează problema ***Decodificare***, adică cea în care se cerea "ghicirea" unei permutări. M-am uitat din nou peste soluția oficială și am văzut că are peste 350 de linii. Orice persoană cu bun simț își dă seama că deja este exagerat să ceri elevilor să rezolve două probleme care, în cel mai bun caz, sunt "muncitorești". Măcar dacă ar fi doar "muncitorești", dar necesită și foarte multe cunoștințe de algoritmică... Ce să-i faci? Dacă vii la olimpiadă, mai trebuie să și suferi puțin!

După ce am rezolvat problema, am privit cu mândrie la numărul liniilor. Erau mai puțin decât 100 și nu peste 350 ca în soluția oficială. Așa că poate era mult mai simplă decât am crezut atunci când am aruncat prima privire peste sursa comisiei! Prezentarea soluției oficiale nu m-a ajutat deloc de data aceasta (este chiar soluția pe care o aminteam la începutul acestui articol). Totuși, la primul punct al rezolvării ceva nu este în regulă. De data aceasta se pare că sunt mai deștept decât comisia, fiindcă algoritmul pe care l-am găsit folosește doar $O(N)$ "întrebări" în loc de $O(N^2)$ pentru a determina poziția corectă a unui element.

Am trecut apoi la problema ***SETI*** și, din nou, m-am uitat mai întâi la numărul de linii al fișierului sursă prezentat de comisie. Avea aproape 200 de linii... Ceva mai scurtă, nu? Dar cam grea și problema asta după numărul de ore de care am avut nevoie ca să o rezolv (șapte). Primul lucru care mi-a atras atenția a fost faptul că, în unele cazuri, trebuie numărate cu-

vintele (subșirurile) de lungime 0. Cam câte subșiruri de lungime 0 credeți că se află într-un șir? Puteți să-mi răspundeți aproape orice și, în majoritatea cazurilor, vă voi da dreptate. Mi se pare absurd să se ceară numărarea subșirurilor vide ale unui șir. Totuși, rulând soluția oficială, m-am "luminat". Vă informez că numărul subșirurilor vide ale unui șir, în concepția autorului problemei, este egal cu numărul literelor din șir! Părerea lui... o respectăm. Totuși cred că este puțin greșită... mai ales că existau șiruri care aveau peste 100000 de litere și în enunț se preciza că un cuvânt *poate avea cel mult* 65535 *de apariții*. Ce să-i faci? Enunțurile se aplică doar în anumite situații... Cu un simplu `if` am rezolvat această dificultate... dar am avut nevoie de sursa comisiei ca să îmi dau seama cum se tratează acest caz. Nu pot decât să presupun (și cam așa ar trebui să fie) că participanții nu au avut acces la această sursă. Chiar dacă unii concurenți ar fi ajuns la concluzia că numărul subșirurilor vide este egal cu cel al literelor, în mod normal ei ar fi renunțat la ea în momentul în care vedeau că numărul total al literelor poate fi mai mare decât 65535. Așadar, pentru a obține toate cele 100 de puncte, trebuia să se ghicească faptul că enunțul mai poate conține și erori...

Cam aceasta a fost prima zi de concurs... Cred că este rezonabil să presupunem că ar fi trebuit scrise peste 1000 de linii de cod; comisia a scris cam 750. Nu pot să nu spun că este destul de clar faptul că membrii comisiei nu au ca scop departajarea concurenților. Ei vor doar să demonstreze că pot inventa probleme pe care nu le poate rezolva nimeni altcineva. Dacă mie mi-a luat, în total, cam douăzeci de ore pentru a rezolva cele trei probleme, nu mă mai mir că marea majoritate a concurenților au avut sub 50 de puncte în prima zi. Totuși, competiția între membrii comisiei trebuie să continue. Nu putem să le "furăm" satisfacția de a spune "*La problema mea nu a luat nimeni 100!*" sau "*La a mea au luat mai mulți 0 decât la a ta!*". Nu știu dacă cineva își aduce aminte de problema de la *ONI* '98 în care, din 50 de puncte posibile, un singur concurent (***eu!!!***) a obținut 10 puncte, altul 5 puncte, iar ceilalți nici un punct! Chiar dacă această extremă nu a mai fost atinsă în următorii ani, se pare că membrii actualei comisiei încearcă să depășească acest record!

Evident, comisia a observat ce s-a întâmplat în prima zi, s-a încercat ca problemele din a doua zi să fie ceva mai simple. Și au fost... La prima problemă (***Suma divizorilor***) sursa comisiei nu are decât puțin peste 750 de linii (ce-i drept cam 700 conțin constante), la a doua (***Sistem***) puțin peste 200, iar la a treia (***Comitat***) tot cam așa.

În plus, cunoștințele de matematică necesare rezolvării primeia dintre aceste probleme sunt cam avansate; din nou, în opinia mea... Am lucrat și aici cam 12 ore până am obținut 100 de puncte la toate aceste probleme.

Rezultatul acestui concurs, previzibil dacă luăm în considerare cele amintite mai sus, a fost că media punctajelor concurenților a fost de aproximativ 60 de puncte. E cam greu la olimpiadă, nu-i așa?

Concluzii

Se observă clar că rezultatele obținute de elevii români la concursurile naționale și internaționale de programare sunt tot mai slabe, chiar dacă toată lumea se laudă cu rezultatele obținute. S-a ajuns în situația ca în ultimii ani să se obțină premii sau mențiuni fără ca vreuna dintre cele șase probleme să fie rezolvată corect. S-au luat mențiuni cu mai puțin de 100 din cele 600 de puncte posibile! Cauzele sunt multiple și nu am eu autoritatea să le analizez. Totuși, este evidentă lipsa tot mai mare de interes a profesorilor și faptul că cele două sau patru ore de informatică din liceele de matematică-informatică nu pot duce la obținerea unor rezultate spectaculoase.

Efectele se văd și pe plan internațional. Până în anul 2000, cu o singură excepție, ***toți*** participanții români la concursurile internaționale (*IOI*, *BOI* și *CEOI*) obțineau medalii. Primul semnal de alarmă a fost tras anul trecut când *România* s-a întors de la *CEOI* 2001 cu doar trei medalii. Anul acesta, din cele 12 medalii "așteptate" a trebuit să ne mulțumim cu șapte.

Cam atât de data aceasta... Aștept reacțiile voastre... Menționez încă o dată că cele precizate în acest articol reprezintă opinii personale.

Medalii...

Olimpiada Balcanică de Informatică (BOI 2002)

Radu Berinde (București), aur

Andrei Benea (Focșani), bronz

Olimpiada de Informatică a Europei Centrale (CEOI 2002)

Victor Costan (București), aur

Daniel Dumitran (București), aur

Olimpiada Internațională de Informatică (IOI 2002)

Radu Berinde (București), aur

Daniel Dumitran (București), aur

Victor Costan (București), argint

Redacția GInfo îi felicită pe toți elevii români medaliați la olimpiadele internaționale de informatică din acest an!

Așteptăm scrisorile voastre!

POȘTA redacției

Natașa Dumitran

Aștept mereu cu nerăbdare ca fiul meu să cumpere sau să primească *Gazeta de Informatică*. De ce? Nu pentru că un colectiv de oameni sufletiști și împătimați în ale informaticii pun la dispoziție elevilor doritori, avizi de cunoaștere, articole care mai de care mai interesante pentru că eu tot nu pricep nimic din ele. Citesc de obicei editorialul, uneori diverse glume care apar și mai sunt și unele noutăți: ce s-a scris, cine a scris, despre ce a scris.

Vă mulțumim pentru interesul acordat revistei noastre, chiar dacă nu citiți chiar tot ce apare în ea.

Dar mai aflu și cum stă informatica românească în momentul *t*, cine a participat, la ce concurs, ce rezultate a obținut. Este acesta chiar un ghid pentru unii părinți în drumul pentru alegerea unui liceu cu profil informatic. Spre surprinderea mea, deși în vara aceasta s-au desfășurat patru concursuri internaționale de informatică (nu este nevoie să le amintesc deoarece au fost publicate în ultimul număr al gazetei problemele cu care s-au confruntat elevii la trei dintre cele patru olimpiade) se pare că România a fost reprezentată de niște anonimi. Este prima oară când sunt ignorați acești tineri și nu mi se pare normal ca tocmai revista *GInfo*, care ar fi trebuit să procedeze ca în anii precedenți, să facă abstracție de cei care au reprezentat cu cinste țara. Măcar celor care sunt preocupați de informatică (și consider că cei care citesc *GInfo* se încadrează în această categorie) să li se prezinte cei care au obținut performanțele deloc neglijabile. La vremea respectivă cotidianele de mare tiraj au prezentat succint aceste rezultate, dar vara *GInfo* este în vacanță. Mi se pare firesc ca ei să fie nominalizați (cum s-a procedat mereu până acum), să se cunoască din ce orașe și de la ce licee provin performerii și cu cine și cum s-au pregătit pentru marea performanță. Oare spațiul este atât de limitat încât nu puteau fi inserate câteva rânduri? Spun asta tocmai pentru că articolul care m-a impresionat se numește *Valoare și respect*. Știu că aveți respect pentru acești elevi muncitori și talentați, dar ei nu simt asta dacă dumneavoastră îi ignorați.

Nu am avut niciodată intenția de a-i ignora pe cei care au obținut rezultate deosebite la olimpiade. Cauzele pentru care numele lor nu au apărut încă în GInfo sunt multiple. În primul rând nimeni nu a considerat că este necesar un comunicat de presă referitor la rezultatele elevilor români la olimpiadele internaționale sau, cel puțin, noi nu am primit nici unul. În al doilea rând, în acest an s-a considerat că nu mai este necesar sprijinul membrilor redacției GInfo în pregătirea elevilor români, motiv pentru care nu am putut afla numele elevilor nici "la fața locului".Nu vreau să găsesc scuze, pentru că puteam încerca să găsim informațiile necesare pe alte căi. Într-adevăr, o mare parte din vină ne aparține. Între timp ne-am reparat (cel puțin parțial) greșeala; pe pagina anterioară se află numele elevilor români care au obținut medalii la edițiile din acest an ale IOI, CEOI și BOI.

De fapt, de mult doream să mă adresez revistei dumneavoastră și acum am găsit prilejul. Poate aducându-vă la cunoștință nemulțumirile mele despre ceea ce se întâmplă azi cu informatica în școli, veți putea contribui la îmbunătățirea situației.

Din păcate vă înșelați! GInfo nu poate să contribuie la îmbunătățirea situației. În ultimii ani, toate sugestiile și opiniile exprimate de redacția GInfo au fost ignorate. Acum câțiva ani opinia noastră mai conta, dar situația s-a schimbat. Motivele ne scapă...

Subsemnata a sesizat presei că în 1999 numărul orelor de informatică predate în licee a scăzut de la opt la trei sau, în cel mai bun caz, la cinci, în cazul claselor intensive de informatică și că, în viitor, rezultatele vor fi firesc direct proporționale cu timpul afectat acestei discipline. Acest lucru s-a adeverit din păcate: în 2001 numărul premiilor obținute de elevii români la olimpiadele internaționale a fost *n* - 1, *n* fiind numărul participanților, în timp ce în 2002 numărul premiilor a fost *n* - 5.

Este și reducerea numărului de ore una dintre cauzele rezultatelor mai slabe obținute de elevii români. Mai sunt și altele, dintre care aș aminti lipsa de interes a profesorilor și, nu în ultimul rând, interesul tot mai mic al elevilor. Învățământul românesc nu trebuie să aibă ca scop doar "producția" de olimpici internaționali. Reducerea numărului de ore nu ar trebui să îi afecteze pe cei pasionați care, oricum, nu învață aproape nimic de la școală. Într-adevăr, nu cred că se poate face serios informatică în trei ore pe săptămână (chiar două în unele licee). Totuși, cele opt sau nouă ore care erau

înainte, erau prea multe pentru un elev obișnuit. Se poate discuta mult pe tema numărului de ore; totuși, alegerea nu ar trebui să se bazeze în nici un caz pe rezultatele de la olimpiadele internaționale.

Pe lângă numărul redus serios de ore de informatică din școală, căci ea oricum nu pregătește elevii pentru performanță căci nu poate, nu are timp, nu are profesori pregătiți în acest scop sau orele de la catedră prevăd o altă programă, pregătirea elevilor din lot s-a făcut în condiții vitrege, poate mai vitrege ca oricând. *ONI* a fost ultima olimpiadă desfășurată, toate celelalte olimpiade "cu ieșire în exterior" având loc în prima vacanță de primăvară. În perioada 21 - 26 mai s-a desfășurat prima perioadă de pregătire a lotului lărgit și al doilea baraj, la *Focșani*, programate, înghesuite și scurtate din cauza sesiunii speciale de bacalaureat care a început pe data de 3 iunie, dar pentru care înscrierile se făceau pe data de 29 mai. Elevii de clasa a XII-a erau evident mult mai stresați decât colegii lor mai mici. A doua perioadă de pregătire și ultimul baraj s-au desfășurat după *BOI* și *CEOI*, la *București*, unde componenții lotului erau nevoiți să se deplaseze cu două mijloace de transport până la locul unde se făcea pregătirea (*Institutul Politehnic București*). Și această perioadă de pregătire a fost scurtată cu cel puțin o zi din cauza bugetului redus al *MEC*. În momentul începerii acestei pregătiri toate loturile restrânse de la celelalte discipline erau de mult stabilite. În același timp trebuie spus că această perioadă a coincis și cu înscrierea la facultate a absolvenților de anul acesta, iar pentru elevii din provincie a fost chiar o problemă.

În opinia mea condițiile în care au loc diferitele concursuri nu pot fi puse pe seama bugetelor reduse, deși acesta este unul dintre factori. Principala cauză este, pur și simplu, indiferența față de olimpicii români, în particular (dar nu în special) față de informaticieni.

Și pentru că atâta stres nu este suficient, finala concursului "Bursele Agora" a fost programată inițial în timpul barajului, iar apoi pe 1 iunie, deci cu două zile înaintea sesiunii speciale de bacalaureat, motiv care a determinat unii finaliști, în special absolvenți ai liceului din promoția 2002, să spună dezamăgiți adio finalei deși ajunseseră aici după o muncă de cinci luni. Ea a rămas nefinalizată. Finala ar fi fost o competiție în plus, o experiență nouă, căci important este să participi.

Nu cred că cineva ne poate acuza că am avut vreun interes să "eliminăm" din competiție anumiți concurenți. Din nou, s-a considerat că redacția **GInfo** *nu trebuie să fie interesată de anumite informații. Am aflat că data inițială în care a fost programată finala se suprapune cu perioada de pregătire a lotului de la concurenți. Am amânat-o cu o săptămână, fără să știm că noua dată este apropiată de cea a începerii sesiunii speciale de Bacalaureat. Din păcate, acestea erau singurele două date în care, din motive întemeiate, redacția* **GInfo** *putea organiza finala, motiv pentru care nu am mai putut modifica data. În săptămânile următoare nu mai aveam acces la laboratoarele de informatică ale liceelor sau universităților clujene deoarece urmau atestatul, admiterea în licee, bacalaureatul etc. Pur și simplu nu am mai găsit un alt sfârșit de săptămână în care laboratoarele să fie libere. Ar fi trebuit să organizăm finala în timpul sesiunii studenților sau a vacanței de vară. În ambele situații, mulți concurenți nu ar fi putut participa fie pentru că ar fi avut examene, fie pentru că ar fi fost în vacanță. Profit de ocazie pentru a-mi exprima o părere personală referitoare la "pseudobacalaureatul" din sesiunea specială. Nu cred că un olimpic are probleme dacă este examinat în același timp cu ceilalți elevi. Am dat și eu un astfel de bacalaureat și cam știu despre ce este vorba. Sesiunea specială ar trebui să fie doar pentru elevii care participă la concursuri internaționale în perioada în care are loc bacalaureatul. Ar fi mult mai elegant dacă olimpicii ar primi bacalaureatul "din oficiu" decât să se organizeze o asemenea "mascaradă".*

Având lângă mine doi mici informaticieni, aflu de la ei multe dintre problemele cu care se confruntă elevii. Deși cel mare a luat patru ani la rând premiul I la *ONI* a fost foarte nemulțumit de noul regulament al *MEC* referitor la premierea participanților la olimpiada națională. La deschiderea olimpiadei, la *Brăila*, fiecare participant a primit o mapă cu instrucțiuni, programul organizat și regulamentul de premiere. În regulament scrie negru pe alb că se vor acorda cinci premii. Mi se pare o nedreptate această micșorare a numărului de premii, căci noi de obicei nu facem altceva decât să ne aliniem la alte state. Dacă la olimpiadele internaționale primii 8% dintre concurenți primesc premiul I, 16% premiul II și tot atâția premiul III de ce nu putem aplica și noi același regulament?

Nu poate fi găsită nici o scuză pentru faptul că regulamentul primit de elevi nu era cel real (a fost anunțat cu destul de mult timp înaintea olimpiadei că vor fi doar trei premii). Referitor la numărul de premii, nu pot decât să citez câteva cuvinte care vă aparțin "important este să participi". Acum, un premiu la națională are o semnificație mult mai mare decât în anii anteriori. Înainte puteai primi premiu sau mențiune chiar dacă nu rezolvai nici o problemă, ceea ce era absurd. La concursurile și olimpiadele sportive se acordă doar trei premii, de ce ar fi altfel la olimpiadele "științifice"? Știu că este cam forțată această comparație, dar ar trebui să ne mulțumim cu ce ni se oferă. E mai bine așa decât să se desființeze olimpiada națională, așa cum s-a întâmplat la alte discipline.

MEC a justificat elaborarea acestui regulament prin faptul că fiind premii mai puține (deci bugetul ar fi de vină) ar avea posibilitatea să acorde premii de valoare mai mare. Acest lucru nu este adevărat și voi demon-

stra acesta. În 2001, obținând premiul I, fiul meu a primit 3000000 de lei și un aparat de fotografiat *Minolta*, în condițiile în care s-au acordat trei premii I și un număr dublu de premii II și III. Anul acesta a primit 3500000 de lei și un colet cu cărți pe care nu le mai citește nimeni: câteva volume de *Panait Istrate*: *Rusia sovietică*... Având în vedere inflația, consider că valoarea premiului de anul acesta este inferioară celei de anul trecut, dar se pare că motivul adevărat este altul.

Într-adevăr, premiile au fost mai mici decât anul trecut, dar mult mai mari decât în anii precedenți. Eu am participat de trei ori la olimpiada națională (în perioada 1996-1998) și am fost premiat cu sume cuprinse între 50000 și 100000 de lei. La valoarea de atunci a dolarului, premiile sunt cuprinse între 8$ și 12$. Pentru participările la olimpiadele internaționale și medaliile obținute am primit, în total, 3.150.000 lei, adică aproximativ 200$. Începând cu anul 2002, informaticienii români medaliați la olimpiadele internaționale au fost premiați cu premii în valoarea de mii de dolari. Vreau să spun că valoarea premiilor este acum mult mai mare, chiar dacă anul acesta face excepție.

S-a micșorat numărul premiilor deoarece se pare că nu convine facultăților să primească un număr mare de studenți olimpici premianți ai olimpiadelor naționale care nu susțin examen de admitere.

Mi se pare normal ca facultățile să nu fie de acord să primească studenți fără examen de admitere. Un olimpic nu ar trebui să aibă probleme la un astfel de examen, dar se pare că mulți au. Cunosc cazuri în care studenți care au intrat la facultate fără admitere datorită unui premiu III la națională obținut în clasa a IX-a sau a X-a au avut restanțe în fiecare an și până la urmă nu au reușit să termine facultatea. La fel ca și "pseudobacalaureatul", nici o astfel de măsură nu mi se pare bine-venită, deși am beneficiat și eu de ea.

Întorcându-ne însă la premiere și la ce scria în regulamentul de premiere primit de fiecare participant la olimpiadă, se vede neconcordanța. S-au acordat trei premii: un premiu I, un premiu II și un premiu III nicidecum cinci. Numai cine accesa *site*-ul *MEC* putea citi că numai dacă numărul participanților depășește 75, numărul premiilor este cinci, omisiune făcută pe foile referitoare la regulamentul primit de competitori. Pe de altă parte, dacă la matematică și fizică (nu mă pronunț în cazul chimiei pentru că nu știu) numărul participanților este totdeauna mai mare de 75, la informatică acest număr nu va fi atins pentru că este limitat de dotarea de care se dispune. Se vor găsi totdeauna 75 de coli de scris pentru matematică, dar nu se știe dacă se vor găsi mai mult de 75 de calculatoare pe de o parte, iar pe de altă parte *MEC* nu va fi dispus să mărească numărul competitorilor, calculele sale raportându-se mereu la cele anterioare.

Neglijențele și aplicarea părților convenabile din regulamente este o practică des întâlnită. Pentru a modifica o astfel de mentalitate trebuie luate măsuri la nivel național. Care ar fi acestea? Nu prea știu fiindcă nu am pregătirea necesară...

Ar fi mult mai multe nemulțumirile mele, dar nu știu care este cadrul în care ar trebui să fie exprimate și cine ar putea ajuta informatica. Cine ar putea ajuta profesorii care-i îndrumă pe elevii performanți? Materia de la școală este una, iar cea pentru competiții iese mult din tiparul acesteia. Oare în condițiile exploziei informaționale, când în fiecare moment apar lucruri noi care revoluționează toate domeniile se mai poate pune problema unor tipare la informatică?

Profesorii trebuie în primul rând să se ajute singuri. E ușor să spui că e prea greu să ții pasul cu tehnologia și să nu faci nimic. Problemele de la concursurile de informatică devin tot mai grele, deci profesorii trebuie să se pregătească și ei pentru a-i putea ajuta pe elevi. Din păcate, tot mai mulți renunță. Evident, cei care nu renunță trebuie recompensați (în primul rând financiar). Astfel de recompense i-ar motiva și pe ceilalți. Momentan, salariile și gradațiile de merit pe care le primesc profesorii (printre altele și datorită faptului că unii dintre elevii lor au obținut performanțe deosebite) sunt cam mici. Există propunerea ca profesorii să fie remunerați în funcție de performanțele elevilor (acestea nu se limitează la rezultatele de la olimpiade). Cred că astfel s-ar rezolva o parte din probleme.

Cei care pregătesc elevii informaticieni sunt niște dascăli deosebiți, dispuși să muncească permanent alături de și lângă discipolii lor. Nimeni nu îi prezintă, nimeni nu-i recompensează și azi din păcate satisfacția că elevii pregătiți de ei au obținut performanțe la nivel internațional nu mai ține din păcate nici măcar de cald. Elevii medaliați la olimpiade au fost nemulțumiți că în 2000 MEN a "uitat" de profesorii care i-au pregătit. Munca individuală a elevilor nu este suficientă. Este nevoie de un suport permanent, de multă experiență, de o dojană și de o încurajare, de cineva care să-ți dea încredere în tine, să te susțină atunci când ți-e greu și să se bucure alături de tine la fiecare succes. Acești oameni, acești dascăli minunați, uneori chiar studenți cu performanțe olimpice, nu au fost prețuiți și mai ales în domeniul *IT* ei își pot găsi refugiul la firme care măcar îi plătesc mai bine. Și cu toate astea ei nu dezertează și "sunt fericiți să se ocupe de elevii doritori".

Îmi permit să vă corectez puțin; doar unii (și destul de puțini) dintre cei care pregătesc informaticieni sunt niște dascăli deosebiți. Aceștia ar trebui recompensați așa cum merită. Din păcate, unii profesori se folosesc de elevii deosebiți "pentru a ieși în față". Am participat, ca membru al comisiei, la două olimpiade naționale și la mai multe baraje de selecție. Am fost profund dezamăgit când am observat care sunt motivele principale pentru care participă anumiți profesori. Certu-

rile pe anumite teme sunt penibile. Toți au dreptate și nu prea îi interesează părerea celorlalți. Cei care nu sunt de acord cu "șefii" sunt imediat dați la o parte. Se creează o "gașcă" de profesori care fac ce vor și rezultatele se văd... Vă recomand să citiți, dacă nu ați făcut-o deja, articolul din acest număr referitor la ONI 2002. Mulți sunt dascăli deosebiți doar atâta timp cât își servesc propriile interese. Adevărații profesori pasionați preferă să rămână anonimi și nu așteaptă să primească recompense mai mari pentru că și-au făcut meseria, deși au făcut-o mult mai bine decât ceilalți.

La fel de mult i-a supărat pe informaticieni faptul că în 2000 excursia olimpicilor internaționali, în *Spania*, a fost programată concomitent cu *CEOI*-ul care s-a desfășurat la *Cluj-Napoca*, iar în anul 2002 excursia din *Italia* s-a desfășurat concomitent cu *IOI*-ul din *Yong-In, Coreea*, informaticienii fiind astfel frustrați de recompensă. A fost nevoie de memorii la *MEC* pentru ca să se înțeleagă că acești tineri trebuie să fie recompensați echitabil chiar retroactiv... zbucium, supărări care puteau fi evitate. În acest moment sunt trei informaticieni internaționali care au fost privați de această bucurie: doi de câte o excursie și unul de două.

Și mai sunt mulți alții... Nici MEC nu poate organiza excursiile olimpicilor astfel încât să nu se suprapună cu nici o olimpiadă internațională sau cu alte activități. Mai sunt și patru informaticieni care au fost privați de un concurs internațional; în 1998 România nu a participat la Balcaniadă datorită unei cereri a Greciei. Așadar, cei patru au pierdut o ocazie (unică pentru unii dintre ei) din motive de politică externă. Este normal ca elevii merituoși să fie recompensați și cu excursii de acest gen, dar nu este o tragedie dacă se pierde o astfel de excursie. De obicei, în anul următor se oferă o "excursie în compensație" (eu am beneficiat de una). Totuși, este regretabil faptul că se promit astfel de premii și promisiunile nu sunt respectate.

Avem semne de la *MEC* că în vara anului 2003 se va rezolva această problemă, dar ea ar fi putut fi evitată.

Este foarte probabil ca evitarea acestei probleme să fi cauzat alte probleme pentru alții. De obicei așa se întâmplă... Faptul că ea va fi rezolvată este, cred eu, încurajator.

Sunt atâtea probleme încât risc să vă plictisesc, dar ele afectează elevii, cei pentru care editați în primul rând această revistă.

Nu ne plictisiți deloc. Aceste probleme există și trebuie discutate. Din păcate nu este în puterea noastră să încercăm să le rezolvăm. Nu putem decât să le semnalăm în paginile revistei. Trebuie totuși să precizez că anumite probleme nu trebuie amplificate sau create acolo unde nu există.

În ceea ce privește respectul pentru profesor el "vine" și din educație. Sunt părinți mulți care-și cresc copiii într-un respect deosebit pentru profesori, conștienți de menirea lor, de dificultatea muncii lor, de rolul lor, dar sunt și mulți părinți care sfidează într-adevăr profesorii și nu-i respectă dând aceeași educație copiilor lor. Aveți posibilitatea, dragi profesori, ca prin prestația dumneavoastră să vă faceți respectați chiar dacă nu există pădure fără uscăciuni.

Părerea mea este că profesorii nu trebuie respectați "din principiu". Într-adevăr, sfidarea profesorilor nu trebuie încurajată de părinți. Există și cazuri în care profesorii nu sunt demni de meseria pe care o practică. Ce părere aveți despre un profesor, membru marcant al comisiei de la olimpiadă, care își ține orele atunci când își aduce aminte că aceasta îi e meserie fiindcă e ocupat cu alte activități "mult mai importante" (după părerea sa)? În plus, la sfârșitul semestrului jumătate dintre elevii săi rămân corigenți pentru a li se demonstra că au nevoie de meditații. Evident, meditațiile trebuie să fie cu profesorul respectiv pentru că altfel vor rămâne din nou corigenți.

Am citit dedicația fiului meu pe fotografia care-l prezenta cu medalia olimpică la gât, dăruită unei profesoare care l-a pregătit acum doi ani. Ea nu denotă deloc lipsă de respect ci dimpotrivă, așa încât vă las pe dumneavoastră să trageți concluzia: ***"Spre amintire, doamnei profesoare Rodica Pintea, cea căreia îi datorez o parte din fiecare premiu obținut și din fiecare excursie în străinătate pe care am efectuat-o."*** Este însă respectul unui elev pentru profesoara lui și veți găsi mulți elevi stimându-și profesorii, apreciindu-i și străduindu-se să învețe și pentru a nu-și dezamăgi profesorii care-și pun multe speranțe în elevii lor. Este o satisfacție mică, dar cred că viața unui profesor care se dedică elevilor/studenților săi este împlinită măcar pentru moment în astfel de ocazii.

Datorită faptului că ați dat un nume, mă văd obligat să îmi exprim și eu o părere referitoare la d-na Rodica Pintea. Este unul dintre puținii profesorii pe care îi cunosc care au avut o contribuție importantă (pentru care, din câte știu eu, nu a cerut recompense de tot felul așa cum fac alți profesori mai mult sau mai puțin direct). Ar trebui să fie un model pentru alți profesori. În perioada în care am făcut parte din lotul de informatică am învățat multe de la dânsa și îi datorez și eu "o parte din fiecare premiu obținut și din fiecare excursie în străinătate pe care am efectuat-o".

Urez revistei ***GInfo***, realizatorilor și colaboratorilor săi multe succese și o viață cât mai lungă. Este un ajutor extraordinar pentru elevi.

Vă mulțumim pentru urări și pentru scrisoare. Mi-ați dat ocazia să îmi exprim unele opinii care, altfel, cu greu ar fi apărut în paginile revistei. Așteptăm și alte scrisori în care să ne spuneți ce așteptați de la noi.

Avem nevoie de criticile cititorilor pentru ca ***GInfo*** *să devină revista pe care o doriți.*

Mihai Scorțaru

În numărul URMĂTOR

Primul număr al noului an va apărea la mijlocul lunii ianuarie. Câteva dintre articolele pe care le veți putea citi după ce vă veți întoarce din vacanță sunt prezentate în continuare...

Pagini WEB cu PHP 4

La cererea cititorilor am reluat seria de articole dedicate creării paginilor Web. În articolele anterioare am prezentat detalii referitoare la HTML, JavaScript și XML. În acest număr am realizat o scurtă prezentare a interpretorului PHP; următorul număr al revistei va conține un articol mai consistent în care vom prezenta mai mult detalii referitoare la acest interpretor.

G.I.E. 2002

La Academia de Studii Economice din București s-a desfășurat un concurs studențesc de programare. Veți putea citi enunțurile celor trei probleme propuse spre rezolvare la această competiție.

Advanced Micro Devices

În cadrul acestui articol vom prezenta un scurt istoric al companiei AMD. Veți putea afla cum a reușit această companie să pună în pericol supremația celor de la Intel pe piața procesoarelor, oferind (la un moment dat) produse mai performante decât cele ale companiei rivale. De asemenea, veți afla cum a reușit AMD să depășească (cu doar două zile) "gigantul" Intel în cursa pentru atingerea barierei de 1 GHz pentru frecvența la care rulează un procesor.

AMD

Bursele Agora

Ați putut citi în paginile acestui număr detalii referitoare la modul de desfășurare a primelor patru runde ale concursului de programare organizat de redacția GInfo. În GInfo 13/1 veți găsi mai multe informații despre următoarele patru runde, precum și enunțurile problemelor de la rundele #09-#12. De asemenea, va fi prezentat clasamentul general după defășurarea primelor opt etape.

Olimpiadele din 2003

Acest articol se dorește a fi o prezentare succintă a celor mai importante competiții internaționale de programare care se vor desfășura pe parcursul anului viitor: Olimpiada Internațională de Informatică din Statele Unite, Olimpiada de Informatică a Europei Centrale din Germania și, sperăm, Olimpiada Balcanică de Informatică din România.

Compresia datelor

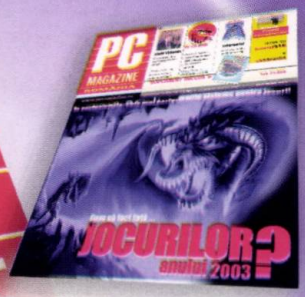
Există două aspecte importante care trebuie luate în considerare la efectuarea unei transmisii de date. În primul rând, datele ar trebui criptate pentru a asigura securitatea transferului. În al doilea rând, pentru a reduce spațiul necesar stocării lor sau pentru a mări viteza transferului, datele trebuie comprimate. În paginile revistei GInfo a fost publicat un serial în care erau prezentate diferite modalități prin care puteau fi codificate datele. În numărul următor va începe un nou serial în care vom descrie diferite metode de compresie folosite pentru a reduce cât mai mult dimensiunea datelor.

Teoreme de programare

Seria de articole dedicată "începătorilor în arta programării" va continua cu prezentarea unei noi teoreme de bază. Va fi descrisă tehnica folosită pentru a selecta toate elementele unui vector care respectă o anumită proprietate dată.

GHIDUL TĂU ÎN LUMEA TEHNOLOGIILOR

PC MAGAZINE ROMÂNIA



Misiunea noastră este de a testa și evalua produse și servicii din domeniul tehnologiei informației și comunicațiilor și de a prezenta rezultatele cu onestitate și obiectivitate

Publicarea acestei reviste este posibilă datorită sponsorizărilor acordate de următoarele firme care s-au decis să sprijine învățământul românesc de informatică

SPONSORI DE PLATINĂ

COMPAQ

SPONSORI DE AUR

FORTE COMPUTERS
Flexibility. Made in Romania

ASTRAL TELECOM

SPONSORI DE ARGINT

s&t
IT SOLUTIONS & SERVICES

hp
invent

ERICSSON

Magua

M.G.T.
EDUCATIONAL

a&C INTERNATIONAL S.A.
the CAD-CAMpany
autodesk
authorized distributor

ALCATEL

*Rugăm companiile care doresc să se alăture acestui efort să ne contacteze prin e-mail la adresa office@ginfo.ro sau la numărul de telefon: 0265-166.516